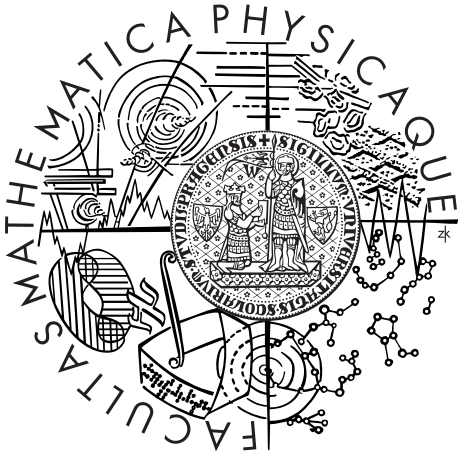
Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Jiří Diviš

Visual Odometry from Omnidirectional Camera

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the master thesis:	Ing. Tomáš Svoboda, Ph.D.
Study programme:	Computer Science
Specialization:	Theoretical Computer Science

Prague 2013

I would like to thank Ing. Tomáš Svoboda, Ph.D. and his team for numerous consults they gave me and for providing the software and the datasets which I used to write this thesis.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In ..... date ..... signature of the author

Název práce: Vizuální odometrie ze všesměrové kamery

Autor: Jiří Diviš

Katedra: Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce: Ing. Tomáš Svoboda, Ph.D.

Abstrakt: V této práci řešíme problém odhadu pohybu robota výhradně z obrázků pořízených ze všesměrové kamery, která je namontována na robotu (vizuální odometrie [SF11, FS12]). V porovnání s hardware běžně používaným pro visuální odometrii, náš robot je specifický tím, že se pohybuje pomocí pásů a obrázky pořizuje pomoví všesměrové kamery s vysokým rozlišením a nízkou frekvencí snímkování (1 to 3 Hz). V naší práci se zaměřujeme na vysokou přesnost odhadů pohybu ve scénách, kde jsou objekty daleko od kamery. Toto je umožněno použitím všesměrové kamery. U tohoto typu kamer je známo že stabilizují odhad pohybu mezi pozicemi kamer, který je špatně podmíněn u kamer s malým zorným polem. Pro odhad pohybu kamery používáme metodu založenou na detekci rohů. K vůli možnosti velké vzájemné rotace kamer mezi snímky jsme nuceni použít metodu párování rohů namísto trackingu.

Klíčová slova: vizuální odometrie, sferická aproximace, odhad pohybu kamery.

Title: Visual Odometry from Omnidirectional Camera

Author: Jiří Diviš

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Ing. Tomáš Svoboda, Ph.D.

Abstract: We present a system that estimates the motion of a robot relying solely on images from onboard omnidirectional camera (visual odometry [SF11, FS12]). Compared to other visual odometry hardware, ours is unusual in utilizing high resolution, low frame-rate (1 to 3 Hz) omnidirectional camera mounted on a robot that is propelled using continuous tracks. We focus on high precision estimates in scenes, where objects are far away from the camera. This is achieved by utilizing omnidirectional camera that is able to stabilize the motion estimates between camera frames that are known to be ill-conditioned for narrow field of view cameras. We employ feature based-approach for estimation camera motion. Given our hardware, possibly high ammounts of camera rotation between frames can occur. Thus we use techniques of feature matching rather than feature tracking.

Keywords: visual odometry, spherical approximation, camera motion estimation.

Contents

Introduction

The subject of this work is the development of visual odometry from omnidirectional an camera for a skid-steer mobile robot.

Visual odometry (VO) is the process of estimating the robot motion using only images from the camera that is rigidly attached to the body. In more detail, given sequence of images ordered by the time they were captured in, it is required to estimate for each of the images the position and orientation relative to each previous image. It is expected that the relative estimate will become gradually less precise as the distance between images increases (it is said that the estimate *drifts*). An excellent introduction to visual odometry is in [SF11, FS12].

Concerning the robot hardware, the robot is equipped with Ladybug3 camera (`www.ptgrey.com`), which is an omnidirectional camera composed of 6 wide-angle perspective cameras. The robot is capable of producing either 3200×1600 panoramic image, or 6 individual images of approx. 1600×1200, both at a frame-rate of 2 to 3 frames per second. The robot is propelled by caterpillar track and it has flipper that can lift front (or back) of the vehicle as pictured in Figure 1.

The motivations for implementing visual odometry on for out robot are the following. The robot already has the capability to perform odometry using other data-sources. Namely, wheel odometry, internal measurement unit and light detection and ranging. The motivation behind having VO for the robot is that it is assumed that it will have greater precision in estimating rotation and that it will work in places where other data-sources fail. Specifically, light detection and ranging based odometry suffers from the fact that laser range is approx. 20 m. The combined wheel and internal measurement unit odometry suffers from wheel slip and rough terrain conditions (high vibrations).

Given the hardware constrains and the variety of odometries on the robot, the design goals of our VO are as follows.

- VO has to be able to reliably estimate motion in an outdoor environment, where it is expected that the objects in the scene will be far away. This goal is motivated by the fact that in indoor environment the light detection and ranging method works well. In outdoor environment this is not the case, because of the range of the method.

- VO has to be able to work with slow frame-rates (compared to 24 hz of standard cameras), but it can take much longer to process frames.

- VO should take advantage of the omnidirectional camera. This is motivated by the fact that omnidirectional vision has been proven in theory and practice to be superior in motion estimation to that of standard perspective camera [Ple03]. The reason for this is that for narrow filed-of-view cameras the estimates of relative motion are ill conditioned.

We will briefly mention several other works that solve similar problem to ours. Perhaps the closest work to our [DRMS07], where motion of a car is estimated using omnidirectional camera mounted above the roof of the car. Other works include [NNB06, DRMS07, HWB+11, SDMK11]. These methods use wide-angle perspective cameras, where some of them perform more difficult problem called

Figure 1: Picture of our skid-steer robot. Image taken from `https://cw.felk.cvut.cz/doku.php/misc/projects/nifti/demos/railway_201204`.

simultaneous localization and mapping. Finally, more unusual methods exists, e.g. [MW08, GAPP07]. For much more through survey of VO, see [SF11, FS12]

1. System Analysis and Design

1.1 Overview

The intent of this section is to provide context for the discussion of both analysis of the VO problem and actual implementation of individual parts of the system. In this section, we describe basic idea behind our implementation of visual odometry (VO) and give an overview of of major components of our system and explain how interact with each other to form the desired behavior. Division into the modules correspond with the actual implementation and thus this section is also an overview for the program documentation[1]. The division is also stable enough to survive severe design changes and it is surprisingly modular.

1.1.1 Simplified Model for Our Problem

In this section we describe what is the desired resulting state of our VO system achieved at the end of successful processing of each image. In further sections, it is then outlined how this is achieved. The state of the system is described in terms of desired model of reality and the parameters of the model that are consistent. The notion of *consistency* will become obvious by the end of this subsection.

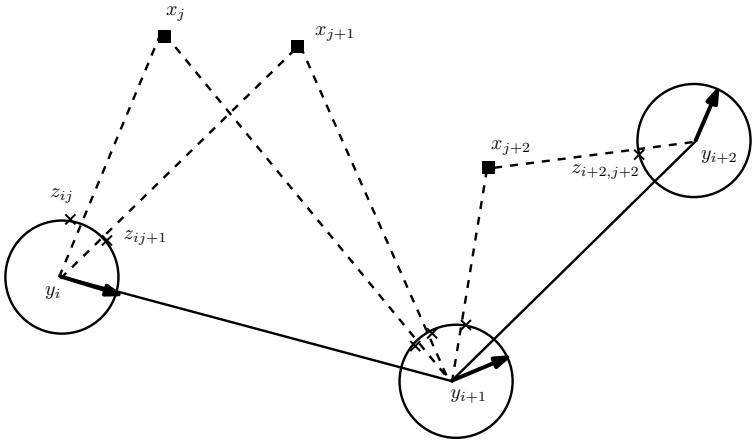


Figure 1.1: Simplified 2D visualization of camera-landmark graph. Black rectangle represent landmarks, the arrows represent camera poses and the circles represent camera imaging surfaces. Dashed lines represent observation rays given by the model and crosses represent constraint given on those observation rays.

We will describe the model in terms of special kind of graph which we call *camera-landmark graph* and there is an example of it in Figure 1.1. In such graph there are two kind of nodes — *landmark nodes* and *camera nodes*. The edges in this graph (represented by the crosses in the figure) represent *constraints*. Constraints are restrictions on the values of otherwise free parameters of the nodes. We explain what the parameters are later in this subsection. Camera node represents the state of the camera at the time given by the time of image

[1]Program documentation is distributed on the DVD which comes with this thesis and is not available in printed form.

capture. *Landmark* is a distinct (in terms of its appearance) part of the scene with appearance that does not substantially change over time. Thus landmark node is a representation of a landmark (black boxes in the figure).

Landmarks are observed via images captured by the camera. Each landmark visible in a camera image has the image of landmark appearance located somewhere on the camera imaging surface (the circles in the figure). The representation of landmark image in particular camera image is called a *feature* (cross in the figure). Landmarks are represented as a set of features which are observations of the landmark and by its position in the 3D space, which is approximated as 3D point. The position of the landmark is free model parameter.

Both landmark and camera nodes (represented as arrows in the figure) in the camera-landmark graph are parametrized by their pose. *Pose* describes the position and orientation of the node entity in the 3D space. This is done formally using *rigid body transforms (RBT)*, which are also called *SE3 transforms*. These are functions which map points from camera coordinate frame (CF) a to camera CF b, which is denoted in this text as T_{ab}. RBTs will not be discussed further in this text, we refer the reader to [MSKS10, Chapter 2].

The observations of a landmark in the camera-landmark graph correspond to the edges in the graph (crosses in Figure 1.1). Given a pose of the landmark, a pose of the camera node and specification of the camera used, it can be determined where in the image plane should the feature corresponding to the landmark observation be located (intersection of the dashed lines and the circles in the figure). It is known that the discrepancy of the actual observation and the predicted position is governed by the normal distribution. Thus the observations of a landmark constrain the pose values of the cameras and the landmark. This explains why the edges in the camera-pose graph are also called constraints.

Our system progressively constructs the model which is represented as camera-landmark graph. From the geometric point of view, the model is determined by the poses cameras and positions of landmarks, landmark observations and by the specification of how camera transfers the incoming light into its imaging surface. The free parameters of the model that are estimated correspond to the landmark and camera poses. If we fix the pose of one camera, and distance of one landmark, it can be shown that with just a few observation per camera[2], there is unique geometric solution for the free model parameters. Of course, if we factor-in the inherent uncertainty of observations, the solution is not unique and more observations per camera can reduce uncertainty in the estimate. To conclude, we note that the camera poses constitute a solution to the VO problem.

1.1.2 The Main Modules of the System

In this section, we describe main modules of the system. The main data structure is described in terms of its function with relation to the rest of the top-level modules.

[2]Each landmark in the graph has to have at least two observations

Information Maintained by the Main Data Structure

We represent data that are required for operation of the VO system as camera-landmark graph. A pose-node of the graph represents a pose of the camera in space as a rigid body transform from origin of the odometry coordinate frame to the coordinate frame of the camera represented by the node. An image of the scene that is taken from this pose is also represented by pose-node — as an array of selected *image features* detectable from this pose (or equivalently, this *viewpoint*). To be more precise, the feature array is are represented as an array of *keypoints* and the corresponding *descriptors* for the keypoints. *Descriptors* contain parts of the feature representation which is useful only for feature matching, while *keypoint* represents information about features that are also useful elsewhere (like image coordinates, or strength of the feature detection response for that feature). The node-node also represents RBT T_wc, where w is the world CF and c is the CF associated with the pose that is represented.

An edge of the graph represents an association of pairs of features in the edge source and edge target that are tentatively considered images of the same *landmark*.

The landmark structure represents 3 things. First, it maintains an association of features-node pairs with landmarks from which they are observed. Second, it maintains whether a meaningful 3D estimate of landmark can be computed. Third, the 3D estimate of landmark position (if applicable). The issue of maintenance of Landmark structure is discussed in section 1.5.

Interface for the Manipulation of the Main Data Structure

The interface consists of routines which create nodes (*Node Builder*), create edges (*Edge Builder*) and delete nodes with consequence of deleting edges (*Node Destroyer*). The addition and deletion of edges invokes changes in the landmark substructure, which is performed by *Landmark Manager* (section 1.5).

Main purpose of Node Builder is to process input image into array of features and its corresponding descriptors. After invocation of Node Builder, the processed image is discarded.

Edge Builder initiates process where most of the work of our VO system is performed. Edge Builder parameters are source and target nodes that are to be connected by an oriented edge. It also accepts the type of edge parameter which determines what kind of estimation is done. Further division of Edge Builder into submodules is discussed in subsection 1.1.3. The arrangement of the submodules that form Edge Builder is than discussed in subsection 1.1.4.

Node Destroyer takes the node that is to be deleted as a parameter. The only interesting issue with this operation is that as a consequence of the node deletion, landmark structure needs to be altered to reflect deletion of features and corresponding landmark-feature associations. This is discussed as a part of Landmark Manager in section 1.5.

Graph Optimalization

Graph optimalization module utilizes the feature-landmark constraints that arise from feature measurements in an image and initial pose estimates of nodes and

landmarks to jointly optimalize estimates of the camera poses. This is a computationally expensive step and it relies on Keyframe Manager to maintain such graph structure in Main Data Structure that keeps only meaningful constraints. Graph optimalization module, that employs nonlinear optimalization method, called *bundle adjustment (BA)* in context of computer vision, is discussed in Section 1.6.

Keyframe Manager

Keyframe Manager is the top level module that directly or indirectly controls all other modules and is an interface of the whole system. Its function is to issue appropriate commands to Node Builder, Edge Builder and Node Destroyer so as to obtain graph structure and thus also node, edge and landmark information that is proper for BA. After BA, pose information of active node is outputed to fulfill VO task. *Active node* is the node that correspond to the last camera image received and processed by Node Builder. The nodes in the graph that are not used in BA are of no further use for VO and thus can be deleted by Node Destroyer. Nodes kept for BA, with the exception of the active node, are commonly called *keyframes* in the context of BA (thus the name Keyframe Manager).

Camera Model

Camera model module is used heavily in almost all modules. Given pose in the scene, its function is to map pixels in the image to the rays that sample parts of scene that determine the value of this pixel and vice versa. Camera model is the subject of section 1.2.

1.1.3 Individual Subcomponents of Edge Constructor Module

We describe submodules from which Edge Builder function is composed. They are described in terms of required input and their output. Because these modules use only information that is part of Main Data Structure, the input requirements also specify which parts of MDS have to be filled for use of this module. The proper order of execution of these modules is then subject of subsection 1.1.4 and Figure 1.2.

Unguided Matching

input Array of keypoints that are to be matched, and array of corresponding feature descriptors.

output Set of pairs (i, j), where i is an index of a feature in the array of features originating from image of source node and j is an index of a feature in the array of target node. It is assumed that if features are paired, that there is reasonable probability p that the paired features are images of the same landmark. Requirements on p are defined by robust estimation module that uses the output, typically p ranges from 0.2 to 0.8).

The task performed by this module is called *feature matching* in general and the pairs (i, j) are called *correspondences*. In more specific terms, the kind of feature matching employed by this module is called *unguided matching* in the sense that no geometric constrains are exploited in matching process and only feature information is used. Unguided matching is the topic of subsection 1.4.1.

Guided Matching

input Array of keypoints that are to be matched and array of corresponding feature descriptors. Rigid body transform between coordinate frames of the two nodes involved in matching. Optionally, landmark-feature association for features in one of the nodes and 3D coordinate of the landmarks involved.

output For *guided matching*, it is qualitatively same as for unguided matching. Guided matching is the topic of subsection 1.4.2.

Robust Model Estimation

input Feature-feature correspondences from two nodes with reasonable number of outliers. RanSaC [FB81] is used for model estimation, and thus outlier to inlier ratio depends on model size, which in turn determines number of iterations required. Computational requirements of various model estimation algorithms are leading factor in time complexity of one iteration.

output The feature-feature correspondences that fit the model and the estimated model — rigid body transform between node coordinate frames. Robust model estimation the topic of section 1.3

Model Refinement

input Rigid body transform estimate (initial model estimate) between the two nodes involved. Feature correspondences that fit the model estimate.

output Refined model estimate (typically using all inlier feature correspondences).

Feature Selection

input Array of keypoints and its corresponding feature descriptors (e.g. such as in a node of the Main Data Structure).

output Array of filtered keypoints and its corresponding array of descriptors. Feature can be selected by non-maxima suppression of feature detection response strength given required maximum acceptable feature density per unit squared of image coordinate.

Outlier Removal Given Estimated Model

input Rigid body transform estimate (model estimate) between the two nodes involved.

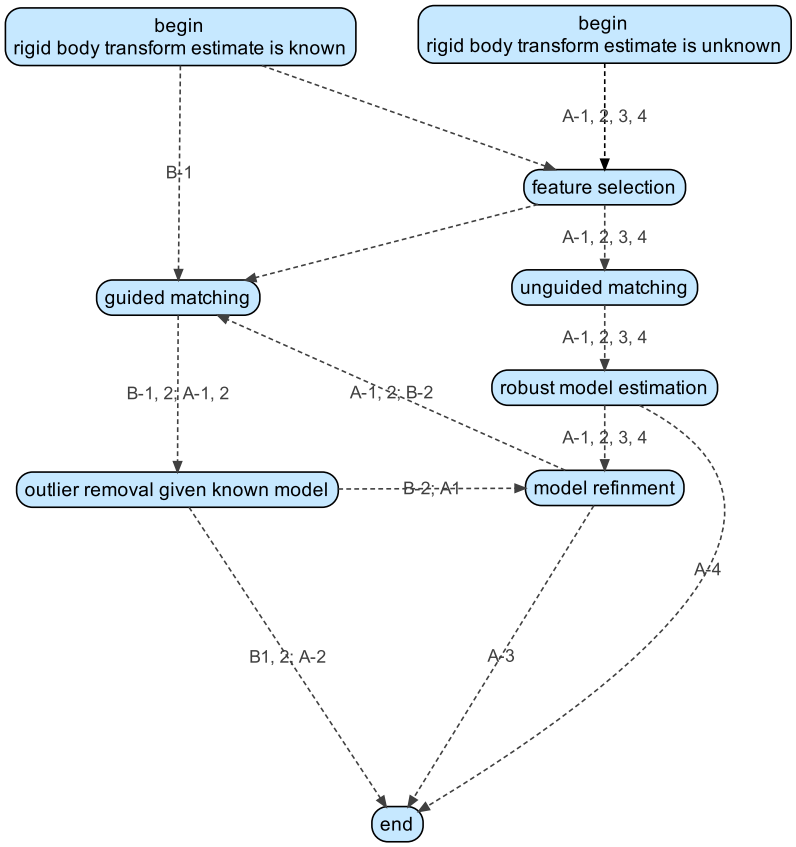


Figure 1.2: Possible arrangements of modules to fulfill Edge Builder interface.

output Feature correspondences that fit the model estimate.

1.1.4 Edge Construction Component Types

We present arrangement of the components from the above section which produces possible implementations of Edge Builder. These arrangement are illustrated in Figure 1.2. They fall into two major types. First, the case where the rigid body transform is not known. Second, the case where the rigid body transform between nodes is already known (e.g. from internal measurement unit).

Unknown Rigid Body Transform

The most complete solution uses the modules in the following order: **feature selection** (to reduce number of features to tractable levels for unguided matching), **unguided matching** (to find feature correspondences), **robust model estimation** (to remove outliers and estimate the rigid body transform), **model refinement** (compute more precise estimate using all data), **guided matching** (to increase number of correspondences), **outlier removal given known model**. This corresponds to path A-1 (this is the case that we use).

Other options include skipping **model refinement** and/or skipping the **guided matching** with **outlier removal given known model** (if we have enough correspondences). This corresponds to paths A-2, A-3 and A-4.

Known Rigid Body Transform

This case is not required for VO to work (and we have not implemented it), but it can help significantly reduce computation time when the rigid body transform estimate is already available. This is useful if VO is combined with e.g. wheel odometry or internal measurement unit. It can be also useful in keyframe manager for creating additional constrains in the pose-graph or in case when only rotational part of the estimate is known.

Desirable arrangement of modules depends on the quality of the initial rigid body transform estimate. If it is fine, only two steps are necessary: **guided matching** and **outlier removal given known model** (B-1). For more coarse estimates, the following steps would become necessary: **guided matching**, **outlier removal given known model**, **model refinement**, **guided matching** (again for more matches), ... (B-2).

1.1.5 Process of Model Construction by the Individual Modules

In this section we describe the sequence of invocations of the previously described modules that produce the above model and estimates its parameters. This is done each time new image is received.

- Node Builder is called on the new images.
- A node in the graph is selected (let it be the previously added node for simplicity) and Edge Builder is run on the two nodes.
  - **Feature selection** is called to select features in the two nodes that will be used for unguided matching.
  - **Unguided matching** is performed.
  - Feature-feature matches from unguided matching are utilized by the **robust estimation** step to compute relative rigid body transform between nodes.
  - Utilizing the estimate and the surviving matches **model refinement** is run to make the estimate of the RBT more accurate.
  - **guided matching** is performed to gain more feature-feature matches that satisfy the model estimated by robust estimation step.
  - **outlier removal is performed on the results**
- new landmarks are initialized and old ones are extended using the **Landmark Manager**.
- Model construction is done. Now its parameters are refined using **Graph Opimalization** module.

1.2 Camera Models

We define *camera model (CM)* using common geometric abstraction used in literature [SRT+11, MSKS10]. Camera model determines which regions of space are imaged on what regions of cameras imaging surface. The regions of scene that affect a point on the imaging surface are modeled by rays. We parametrize *Ray* as (B^f, B^i), where B^f is some point on the ray in 3D space and B^i is some direction vector of the ray. *Camera model (CM)* is fully determined by the following:

- space of its imaging surface $\mathcal{I}$
- the space $\mathcal{R}$ of rays sampled by the camera.
- camera parameters p (*intrinsic camera parameters*).
- one to one map $\pi_p \colon \mathcal{R} \to \mathcal{I}$ between the space of imaging surface $\mathcal{I}$ and of the camera and the space of rays sampled by the camera $\mathcal{R}$.

  The function π_p is called *forward projection* or simply *projection* and the function π_p^{-1} is called *back-projection*.

CMs can be divided into central and non-central. CM is *central camera model*, if all the rays in $\mathcal{R}$ intersect at one point which is called *optical center*. Otherwise it is *non-central camera model* or equivalently *generalized camera model*. Consequently rays in central CM model can be parametrized using direction vectors B^i only, because B^f by convention, lies at the origin of camera CF.

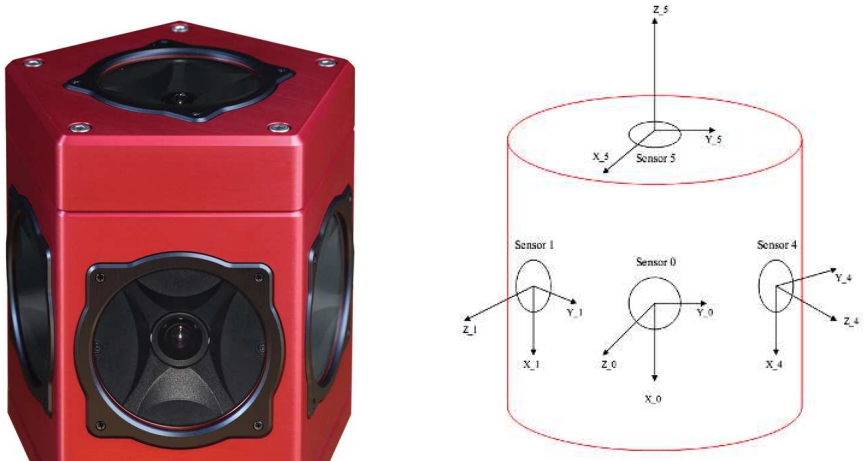
In order for function π_p to be determined, camera parameters p have to be determined. Process of estimating camera parameters is called *calibration*. Designing CM and calibrating it is not subject of this thesis. We were provided few options for CM. In the following text, the available camera models are discussed. The decision to use a one of the models is than discussed in Section 1.3.

1.2.1 Generalized Model of Ladybug3 Camera

Ladybug3 camera consist of six standard cameras with fixed intrinsic parameters adhering to the ideal pinhole CM with polynomial distortion, similar to one described in [MSKS10, Chapter 3]. Five of these cameras are arranged so that their optical axes lie in one plane and meet at one point O_s. Sixth camera has optical axis perpendicular to the plane and also intersects the other optical axes at point O_s. The arrangement of cameras in Ladybug3 camera is depicted in Figure 1.3b, while the picture of the real camera is in Figure 1.3a. The cameras have wide field of view so that all rays with direction vectors pointing above (as defined by Figure 1.3) the plane defined by the optical axes of the first five cameras are sampled by at least one of the cameras (or more).

The optical centers of the individual cameras do not coincide with O_s. Therefore, if we wish to model Ladybug3 camera by single CM precisely, that CM cannot be central CM. We designate the optical centers of the individual cameras as $O_0, \ldots, O_5$ and describe appropriate CM for Ladybug3 camera as follows.

- The virtual imaging surface $\mathcal{I} = \bigcup_{j=0,\ldots,5} \mathcal{I}_j$ of Ladybug3 camera consists of the six parts that correspond to imaging planes $\mathcal{I}_j$ of the individual cameras satisfying pinhole CM.



(a) Ladybug3 camera.

(b) Ladybug3 camera. Orientation of optical axes of individual perspective cameras.

Figure 1.3: Ladybug3 Camera. Images taken from `www.ptgrey.com`.

- The space of rays sampled by the Ladybug3 camera is

$$\mathcal{R} = \bigcup_{j=0,\dots,5} \{(O_j, B_j^i) | B_j^i \in \mathcal{B}_j\}$$

  where $\mathcal{B}_j$ is the set of rays sampled by the camera with optical center O_j.

- The Ladybug3 camera parameters p are given by:
  - RBTs from camera CF with origin at O_s and CF of individual cameras with origins at O_j.
  - The parameters p_j of the individual pinhole cameras.

  Camera, which consists of multiple cameras related by RBTs is called *multi-camera rig*.

- The forward projection $\pi_p \colon \mathcal{R} \to \mathcal{I}$ is defined by:

$$\pi(O_j, B_j^i) = \pi_{p_j,j}(B_j^i)$$

  and back-projection is thus defined by:

$$\pi_p^{-1}(I_j) = (O_j, \pi_{p_j,j}^{-1}(I_j))$$

1.2.2 Central Camera Models for Omnidirectional Cameras

In this section we define two useful central CMs which will be useful for the next section where we consider an approximation of Ladybug3 CM using these models.

Spherical Camera Model

Convenient model applicable to all central projection cameras is *spherical CM* which is defined as follows:

- The virtual imaging surface $\mathcal{I}$ is a unit sphere.
- Space of rays $\mathcal{R}$ samples visible rays B^i with optical center O_s.
- The forward projection $\pi_p \colon \mathcal{R} \to \mathcal{I}$ is defined by

$$\pi_p(O_j, B^i) = (x, y, z)$$

  where (x, y, z) is the direction vector of a ray B^i such that $\|(x, y, z)\| = 1$.
- The back-projection π_p^{-1} is identity.

Cylindrical Camera Model

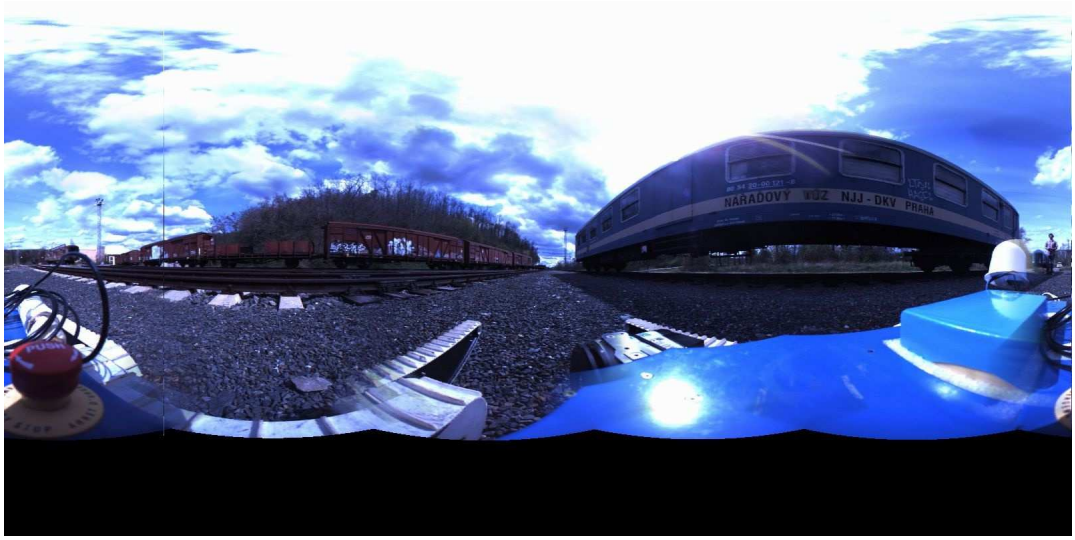


Figure 1.4: Panoramic image constructed from spherical approximation.

This is a central CM, where rays are projected onto a cylinder which is then unrolled into a plane. One of the features of this model is that as the elevation of the projected ray increases the angle between the rays corresponding to two equidistant points on the image plane also increases which causes severe distortion on the upper and lower edges of the imaging surface (as seen in Figure 1.4).

- imaging surface is defined by $\mathcal{I} = (\theta, \varphi)$, where $\theta \in (-\pi, +\pi)$ and $\varphi \in (-\frac{\pi}{2}, +\frac{\pi}{2})$.
- the space $\mathcal{R}$ is defined as:

$$\mathcal{R} = \{(0, (x, y, z)) | \|(x, y, z)\| = 1 \wedge |z| \neq 1\}$$

- intrinsic camera parameters p are the same as for spherical approximation.

- The projection functions correspond to conversion between spherical and cartesian coordinates. More specifically, forward projection $\pi_p \colon \mathcal{R} \to \mathcal{I}$ is defined as

$$\pi_p((x, y, z)) = (\text{atan2}(-x, z), \text{atan2}(y, \sqrt{x^2 + z^2}))$$

and back-projection is

$$\pi_p^{-1}((\theta, \varphi)) = (\sin(\theta)\cos(\varphi), \sin(\varphi), \cos(\theta)\cos(\varphi))$$

1.2.3 Spherical Approximation for Ladybug3 Camera

If we use notation introduced in the definition of Ladybug3 CM, than in *spherical approximation* of Ladybug3 camera the optical center O_s is assumed to coincide with optical centers O_j. Consider a ray $(O_s, B^i) \in \mathcal{R}$ such that there exists camera j and ray $B_j^i \in \mathcal{R}_j$ for which $B^i = B_j^i$. In the approximation the rays B^i and B_j^i sample sample same part of the scene and thus it is assumed that

$$c(\pi_p(B^i)) = c_j(\pi_{p_j,j}((O_j, B_j^i)))$$

where c, c_j are functions that map point on imaging surface to its brightness value.

In the following, we attempt to characterize error induced by the spherical approximation. We follow the derivation presented in [KHK10]. Consider a landmark with coordinates X that is imaged by ray B_j^i of camera j. In coordinate frame of Ladybug3 CM, this corresponds to the ray (O_j, B_j^i). Using a spherical CM, we designate the ray corresponding to the landmark X as B^i. Such situation with $j = 1$ is drawn in Figure 1.5a. The rays (O_s, O_j), B^i and (O_j, B_j^i) determine a plane, which is drawn in Figure 1.5b. In the following, the approximation error is characterized using angle between rays (O_j, B_j^i) and B^i which is designated as θ_ϵ. This is then related to angle θ_ϵ' which is maximum angle such that error of approximation on the imaging surface of spherical CM, which is determined by $e_j := \|x_s - x_m\|$, does not exceed k pixels.

The derivation of θ_ϵ follows. We designate the distance of landmark X from optical center O_s as d_r and we designate t_j as $\|O_s - O_j\|$. For Ladybug3 CM, $t_j = 0.042\,\text{m}$ for horizontal cameras (cameras $0 - 4$) and $t_5 = 0.062\,\text{m}$ for the top camera (camera 5). Using the law of sines, we get

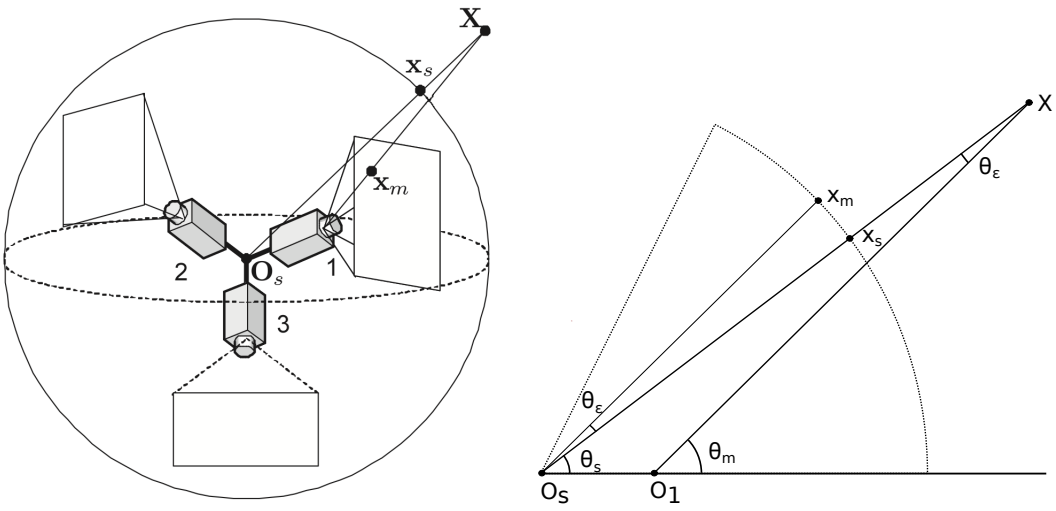
$$\frac{t_j}{\sin\theta_\epsilon} = \frac{d_r}{\sin(\pi - \theta_m)}$$

noting that $\theta_m \leq \pi$, we have $\sin\theta_m = \sin(\pi - \theta_m)$ an thus

$$\frac{t_j}{\sin\theta_\epsilon} = \frac{d_r}{\sin\theta_m}$$

For minimal angle θ_ϵ' between the rays intersecting pixel boundaries, we get following condition on d_r, t_j and θ_m

$$\sin\theta_\epsilon' \geq \frac{t_j}{d_r}\sin\theta_m$$



(a) Error in ray direction induced by spherical approximation. Image adapted from [KHK10].

(b) Detail of spherical approximation error (see the text).

Figure 1.5: Spherical approximation.

This means that for specified error of approximation θ'_ϵ, maximal angle $\theta'_m = \max \theta_m$ and distance $\|O_i - O_s\| = t_j$, the minimal distance of the scene d_r has to be:

$$d_r \geq \frac{t_j \sin \theta_m}{\sin \theta'_\epsilon} \tag{1.1}$$

Because we use spherical approximation with panoramic image at a resolution of 1600×800 px, we now analyze particular values of d_r and θ'_ϵ that apply for this model. Unfortunately, θ'_ϵ is not even approximately constant in panoramic image. In order to illustrate the nature of the error, we plotted θ'_ϵ for each pixels in panoramic image (see Figure 1.6). The formula for computing $\theta'_\epsilon(x, y)$ is as follows

$$\theta'_\epsilon(\theta, \varphi) = \min_N(\|\|\pi_p^{-1}((\theta, \varphi))\| - \|\pi_p^{-1}(N((\theta, \varphi)))\|\|) \,,$$

where π_p is a projection function for panoramic CM and $N((\theta, \varphi))$ are image plane coordinates of the neighbouring pixes of (θ, φ).

To arrive at an estimate of acceptable scene distance, we plot d_r as a function of θ_m, which is determined by Equation 1.1. We plot this for $\varphi = 60°$, because images of objects with higher elevation are too distorted to be useful for reliable feature detection and they are usually far away anyway. The plot is presented as Figure 1.7. We also note that for pixel neighborhood function N in vertical direction only, $\theta'_\epsilon(\theta, \varphi) = \theta'_\epsilon(0, 0)$. For this θ'_ϵ, the plot of d_r as a function of θ_m is in Figure 1.8

With these values, the spherical approximation does not seem to be plausible. However, the results presented in [KHK10] indicate that it should be possible to reliable use the spherical approximation for scenes where most objects are further than 5m. This does not apply to the bottom half of the image because it is mostly covered by robot and we do not use it.

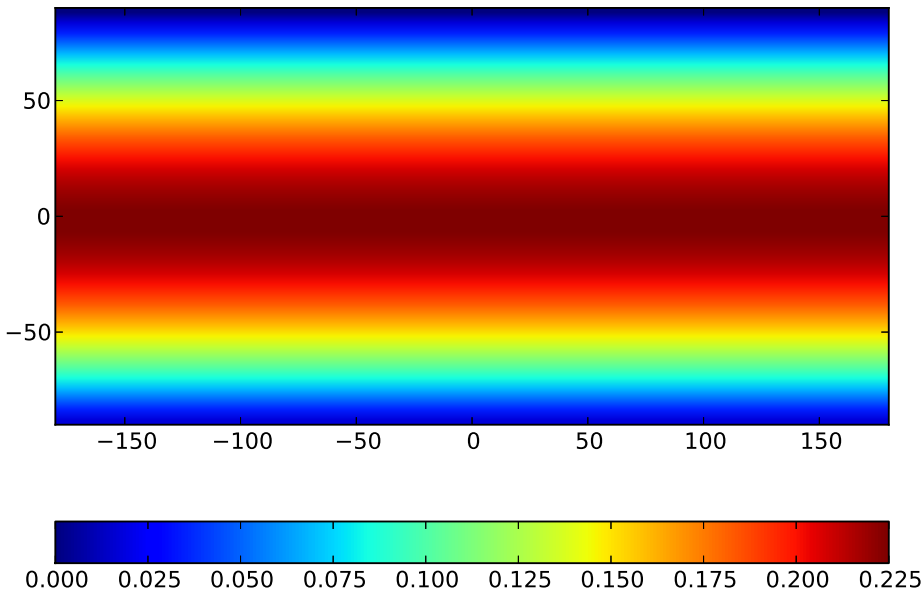


Figure 1.6: Plot of θ'_ϵ for the panoramic image plane. The values are in degrees.

1.3 Estimating Relative Rigid Body Transforms Between Two Cameras

In this section, we discuss the problem of computing rigid body transform (RBT) between the coordinate frames of two nodes in the pose-graph from set of feature-feature or feature-landmark associations (or both), where we allow the associations to be partially incorrect. We call this problem *pose-graph edge transform problem*. All considered methods for solving this problem require camera model with backprojection function. Part of the problem of choosing the method is also the choice of appropriate camera model for our physical camera as the set of usable methods depend on the camera model used.

More precise definition of *pose-graph edge transform problem* is as follows. We have pose-graph in state where there is a node n that is not contained in any pose-graph edge and a pair of nodes (n, n_p). Node n does not have an estimate of the RBT T_{nw}. We wish to make an initial estimate of T_{nw} that will later be further optimized using bundle adjustment (Section 1.6). In addition to the data available in the main data structure, we have the set of the feature-feature associations between the nodes in the pair (n, n_p). This has been obtained in the step of unguided feature-feature matching that is described in Section 1.4.1. Note that, we do not consider the case when n and n_p are in different connected components of the pose-graph. The described method could be extended to efficiently allow for this, but we do not discuss it. The reason for this is that this case is not useful for pure VO.

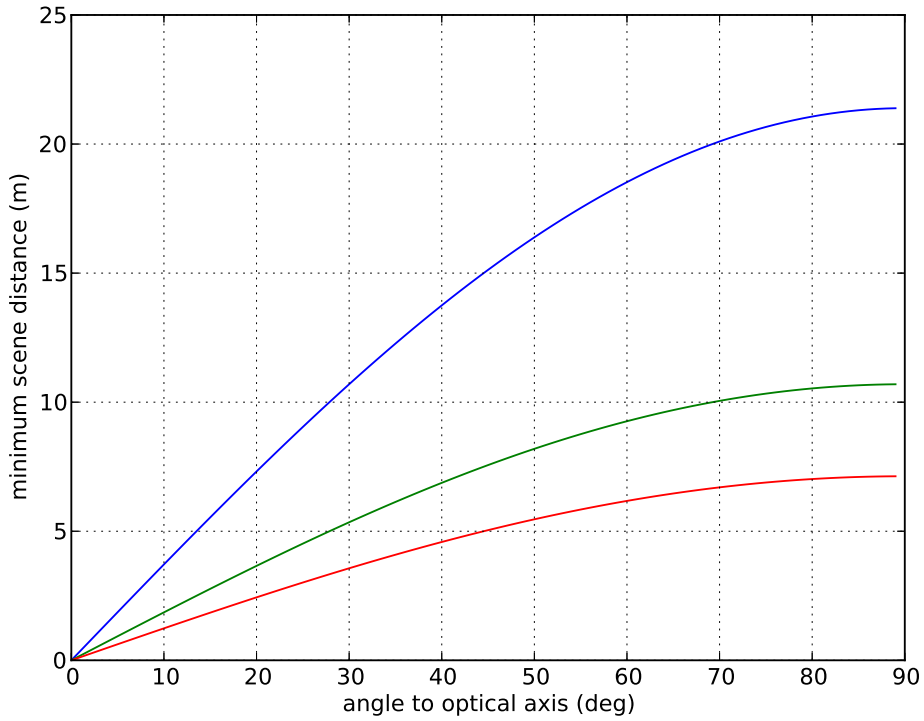


Figure 1.7: Plot of minimum distance to the scene as a function of the angle between optical center and imaged ray at $\varphi = 60°$. The plots correspond to errors k of one, two and three pixels (colors blue, green and red).

1.3.1 Model Estimation in Presence of Outliers

Our problem can be solved by model parameter estimation paradigm [HZ04, Chapter 3]. That is we have model that is an adequate description of reality. Such model is parametrized by a set of parameters. When the parameters of the model are known, we say that the *model is instantiated*. We observe the reality by collecting a set of observations. The observations are governed by observation model that is fully determined by the instantiated model. The procedure used to instantiate model from the observations governed by an observation model is called *model estimator*.

In *robust estimation* of the model it is further possible that observations are polluted by *outliers*, which are observations that are not determined by an observation model. The observations that are determined by the model are *inliers*.

To illustrate the introduced notions, consider the following simple example of estimating a circle in 2D space that is known to be centered in the origin from set of point observations. The model is then the circle itself parametrized by its diameter r. Suppose that we can measure circle by noisily observing a large set of points on the circle. The appropriate measurement model would than be that the observation is a random sample from 2D normal distribution. The model estimator would than estimate the model by taking single datapoint and computing its distance from the origin.

To deal with possible incorrect observations, we employ the method called Random Sample Consensus (RanSaC) [FB81]. This method is de-facto standard

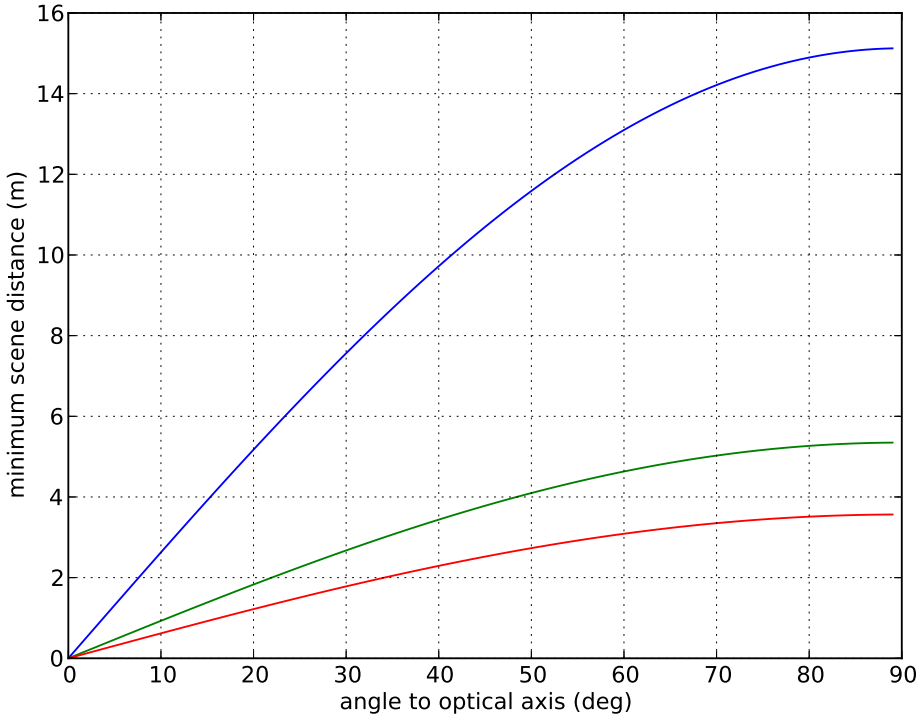


Figure 1.8: Plot of minimum distance to the scene as a function of the angle between optical center and imaged ray at $\varphi = 0°$. The plots correspond to errors k of one, two and three pixels (colors blue, green and red).

for model estimation in presence of outliers [FS12]. In a nutshell this method works as follows. As an input, it takes a set of observations of a model (or *datapoints*), a model estimator procedure and an *inlier function* that determines for each datapoint and a given estimated model a subset of datapoints that are considered inliers. The method than operates by iteratively, by estimating the model from randomly chosen subset of set of datapoints and using inlier function to compute set of inliers. The chosen set is always of size k, where k is constant given by the model estimator. After predetermined number of iterations algorithm terminates by returning the model estimate that has the larges set of inliers. Such model is then accepted, if the largest set of inliers has large enough cardinality. The minimal cardinality of the set is typically determined by expected inlier to outlier ratio for datapoint set. The precise formulation of the algorithm is presented in Algorithm 1. The discussion of choosing appropriate model and inlier function is the topic of subsequent sections.

Number of RanSaC Iterations Required

We proceed by discussing estimates of number of iterations required for the algorithm to succeed (i.e. return a model estimate). It can be shown [FB81], that if we want the algorithm to succeed with probability z, than under the assumption that model size is k and probability of datapoint is considered an inlier by inlier function is w, the required number of iterations n is governed by the following

Algorithm 1 Random Sample Consensus (RanSaC) [FB81]

Computes model estimate M^* that has largest consensus set C^*. As an input it takes model M, set of datapoints P, inlier function $f : M(S) \times P \rightarrow 0, 1$ and a threshold t. $M(S)$ is model instantiated from $k(M)$ datapoints, where $k(M)$ is minimal number of points required to compute unique model.

1. Set $C^* \leftarrow \emptyset$.
2. Randomly select $k(M)$ datapoints from P and denote them S.
3. instantiate model $M(S)$ (using the selected datapoints S).
4. Determine subset C, $C \subseteq P$ that satisfy error constraints given by the inlier function f.
5. If $|C| > |C^*|$ then set both $C^* \leftarrow C$ and $M^* \leftarrow M$.
6. If the desired number of iterations n has not been reached, continue with step 2.
7. If $|C^*| > t$ return failure otherwise return success, model M and consensus set C^*.

formula:

$$n = \frac{\log(1 - z)}{\log(1 - w^k)}$$

This means that order for an model estimator to be usable in a RanSaC scheme, it has to be able to instantiate models fast (i.e. $\gg 1000$ models per second), it has to be able to instantiate from small number of datapoints (i.e. < 10) and there must also exist fast inlier function for the model. A useful point here is that the model estimator does not estimate the model precisely and thus the w can be actually be lower (and typically is) than true probability of obtaining the correct correct observation. This and the fact that model is computed from small number of datapoints are also the reasons why the RanSaC procedure is usually followed by computing a model estimate from all observations and readjusting the inlier set accordingly.

1.3.2 Models and Estimators

In this sections we discuss various models and estimators that are applicable to our problem and that can be used in a RanSaC scheme. All considered options are then presented in Table 1.1. We also discuss how to extract the solution to our problem, if it is not obvious. Finally, we touch on how to deal with problem where the methods presented produce more than one solutions. The inlier are described in the next section.

Table 1.1: Possible methods that can be useful in solving pose-graph edge transform problem. The number k is the minimum datapoints required to compute the estimate and the number l is the maximum number of solutions obtained using the solver

Name	**Model**	k	l	**Camera**	**Code**	**Reference**
Perspective Three-Point	T_{cw}	3	4	central	yes	[KSS11]
Generalized Perspective Three-Point	T_{cw}	3	8	generalized	no	[NS07]
5-Point Relative Pose	E	5	10	central	yes	[LH06]
Generalized 6-Point Relative Pose	T_{cw}	6	64	generalized	yes	[SNOÅ05]

Pose Estimation with Central Cameras

In this section we discuss a method for directly estimating a camera pose T_{nw} based on feature-landmark correspondences. We recall from the beginning that n is an unconnected node and (n, n_p) an edge that is being added to the pose-graph. The method discussed here is commonly called *Perspective from n Points* or *Perspective n-Points (PnP)*. More precisely, the datapoints that are input to the method are pairs (p_i, B^i), where p_i are positions of landmarks l_i with respect to world coordinate frame and B^i are rays that correspond to observations of l_i from pose represented by node n. The tentative landmark-feature associations are computed from feature-feature associations that were computed for the edge (n, n_p).

The PnP problem is known (see papers referenced in [KSS11]) to require at least 3 datapoints of the above kind in order to constrain T_{nw} to finite number of possibilities, of which there are 4. There exist many methods for solving P3P. For all of them, we cite the [KSS11] since it is the most new development in P3P, it claims to be very fast and there is fast C source-code available based on this paper at `http://www.asl.ethz.ch/people/kneipl/personal/p3p_code_final.zip`.

Relative Motion Estimation with Central Cameras

In this section, we discuss case of estimating RBT from two cameras and tentative sets of feature-feature associations. The material that follows is adapted from [MSKS10, Chapter 5] unless otherwise stated.

We begin our discussion by describing the geometric relationships of geometric objects involved. We consider non-degenerate situation of two cameras c_1 and c_2 observing single landmark modeled by point p. The lines (c_1, p) and (c_2, p) correspond to the rays B_1^i and B_2^i sampling point p. Similarly, I_1 and I_2 are projections of point p on the cameras image plane. The plane determined by points c_1, c_2 and p is called *epipolar plane* and the curve determined by the projections of all rays perceived by the camera c_1 that are in the epipolar plane is called *epipolar curve* for camera c_1 and point p. In the case of planar imaging surfaces as in the figure, it is actually a line.

It can be shown that the RBT $T_{c_1c_2}$ cannot be estimated with central CM when landmark positions are unknown. What can be estimated is *essential matrix* E,

which is fully determined by RBT $(R_{c_1c_2}, t_{c_1c_2}) \in SE(3)$ as follows

$$(\exists c \in \mathcal{R})(E = c\widehat{t_{c_1c_2}} R_{c_1c_2})$$

where for given two vectors a and b, $\widehat{a} \in R^{3\times 3}$ is a matrix such that $\widehat{a}b = a \times b$ (cross product of a and b). The E than determines only $(R_{c_1c_2}, d_{c_1c_2})$ where $d_{c_1c_2}$ is related to $t_{c_1c_2}$ as follows. There exists $\lambda \in \mathbb{R}^+$ such that $\lambda d_{c_1c_2} = t_{c_1c_2}$.

The matrix E can be computed from feature-feature associations. The process of doing that is called *relative motion estimation*. It can be shown (see references in [LH06]) that minimal number of such associations required to constrain E so much so that there is finite number of satisfactory essential matrices is 5. This number of feature-feature associations can than guarantee maximum of 10 solutions. A method that estimates E from minimal number of correspondences is described in [LH06] and fast implementation in C language is available on-line at `http://users.cecs.anu.edu.au/~hongdong/index.html`. Other methods exist, but we tested it and found in to this code to be easy to use and good enough for our problem.

Algorithm 2 Recovery of rotation and translation direction from an essential matrix.

Given an essential matrix E corresponding to the unknown RBT T_{nn_p}, extracts rotation component R_{nn_p} and translation direction vector d_{nn_p}.

1. Compute singular value decomposition of matrix, i.e. compute U, S and V such that $E = USV^T$.
2. If $\det(U) < 0$, then set $U \leftarrow -U$
3. If $\det(V) < 0$, then set $V \leftarrow -V$
4. Compute four solutions possible solutions for (R_{nn_p}, d_{nn_p}) as

   $$(\pm UR_Z(+\frac{\pi}{2})SU^T, UR_Z^T(+\frac{\pi}{2})V^T)$$

   and

   $$(\pm UR_Z(-\frac{\pi}{2})SU^T, UR_Z^T(-\frac{\pi}{2})V^T)$$

   where

   $$R_Z(\pm\frac{\pi}{2}) = \begin{pmatrix} 0 & \mp 1 & 0 \\ \pm 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

5. Disambiguate solutions by checking if points are in front of camera as described in the text. In this specific case it is guaranteed that after the disambiguation only one solution remains.

In order to apply relative motion estimation to our problem, we have to have a method of extracting $(R_{c_1c_2}, d_{c_1}c_2)$ from E and we also have to have a way of computing unknown λ by other means. Algorithm 2 does the former. The latter can be accomplished by method suggested in [SF11] as follows. An edge in the pose

graph that contains node n_p is selected. Let the second node in the selected edge be n_s. From each pair of feature-feature matches that shares common features at node n_p, landmarks that correspond to the pairs are triangulated (Algorithm 3) and the ratios of landmark depths wrt. to node n_p are stored. The correct λ is then computed using the median of all stored ratios. If it is the case that n_p and n are the only nodes in the pose graph, we select λ arbitrarily (this only happens once at the beginning). Also care has to be taken in handling cases where n and n_p are close, we discard such nodes n.

Models and Estimators for Non-central Cameras and Others

For both pose estimation and motion estimation problems, there exist algorithms that directly compute T_{nw} or T_{nn_p} respectively. Their properties are summarized in Table 1.1 together with one all other applicable method that computes model from feature-feature association across 3 poses. For one of them, the motion estimation with generalized camera, we found a source code available at `http://vis.uky.edu/~stewe/code/generalized_camera/`.

Solving Issues with Multiple Solutions

All of the above problems produce more than one solution. We discuss three solutions to this problem that should be tried in order they will be presented in. First solution is to triangulate the landmarks from the corresponding observation rays involved in the datapoints from which the solution was obtained and the computed relative camera poses. From the triangulated points it can be then checked if the computed triangulations are in front of the camera (i.e. are in the direction given by observation rays).

The second solution is to use additional datapoint for estimation of the model to test if it is consistent with the solution. That is to test if it is an inlier and than test if it is in from of the camera. Some of the methods (e.g. [LH06]) actually guarantee that this is enough to find an unique solution.

The third solution is to test each of the solutions as a separate hypothesis in the RanSaC algorithm. More precisely, in each iteration of the RanSaC, for all estimates the inlier set is calculated and the one with largest inlier set is kept as a result for the iteration.

1.3.3 Datapoint Error Functions For Rigid Body Transform Models

In this section, we present three commonly used options for the RanSaC inlier function from [MSKS10].

Reprojection Error

This is general method that applies to measuring if a landmark observation is an inlier in the geometric model we have chosen for visual odometry problem (the model was described in Section 1.1). Given a camera pose c and the position of any given landmark position $p(l)$, we can test if the observation $o(c, l)$

Algorithm 3 Landmark Triangulation from Two Observations

Let x_a and x_b be rays that observe landmark l from cameras c_a and c_b. Let $T_{ab} = (R, t) \in SO(3)$ be the rigid body transform that transforms scene points with coordinates in coordinate frame given by c_a into coordinates in coordinate frame given by c_b. Given known x_a, x_b, R and t, the depth estimate λ_a of the landmark l in camera c_a is obtained as follows.

i Construct matrix $M = \begin{pmatrix} \widehat{x_2} R x_1 & \widehat{x_2} t \end{pmatrix} \in \mathbb{R}^{3\times 2}$.

ii Construct column vector $\lambda = \begin{pmatrix} \lambda_a & \gamma T \end{pmatrix}^T \in \mathbb{R}^{2\times 1}$.

iii Compute least square estimate of λ in the equation $M\lambda = 0$.

iv For rigid body transform (R, t), the depth estimate λ_a is then computed as λ_a/γ.

associated with the landmark l in camera c is an inlier by measuring reprojection error. *Reprojection error* is defined as $\pi^c(r(p(l))) - o(c, l)$, where $r(p(l))$ determines ray that samples point $p(l)$. With landmark observation error being normally distributed, we can measure the probability that observation $o(c, l)$ is an inlier using Mahalanobis distance [HZ04]. In the simple case where covariance matrix is $\Sigma = cI$ for some constant $c \in R$, the probability depends on geometric distance $\|\pi^c(r(p(l))) - o(c, l)\|$ and an appropriate value is typically determined experimentally.

The reprojection error inlier function is suited for PnP problems, because the estimate of the landmark position is directly available. In the case of relative motion estimation this is not the case. The problem with solving this by simply computing the triangulation is that it is slow. This is the motivation for the following approach.

Deviation from Epipolar Plane

This is special case where relation between two cameras is specified by an essential matrix E and datapoints are feature-feature correspondences. Consider essential matrix E and a datapoint (x_1^l, x_2^l), where x_1^l is an observation of some landmark l in the first camera and x_2^l is observation associated with the same landmark in the second camera. It can be shown that given E and x_1^l, the ray corresponding to x_2^l is constraint to lie in the epipolar plane that is given by E and x_1^l. Closed form solution for computing an angle between ray corresponding to x_2^l and the epipolar plane exists and it takes only few instructions to compute.

More correct solution would be to measure distance to epipolar curve in the imaging surface. This method has similar statistical interpretation as the reprojection error. The problem with this is that we do not know how to efficiently compute this distance. This method is commonly used in the case of planar imaging surfaces where it can be computed easily, and fast. In case of precise Ladybug3 CM, this method is also applicable to generalized motion estimation – it can be used between pars of separate cameras.

1.3.4 Implemented Solution and Concluding Remarks

We implemented motion estimation using RanSaC scheme with essential matrix for central CM as the model that was estimated. We used 5-point minimal solver for central CM described in [LH06]. As an error function for datapoints, we use the one based on angle between the epipolar planes determined by the associated features that was described above. The method for scale estimation and method form extraction of rotation and direction vector from essential matrix is also implemented as described above.

In the following, we explain our decision to use spherical approximation and our choice of method for relative motion estimation. The of choice of panorama as virtual imaging surface was due to simpler implementation.

Choice of Central Camera Model over Non-central Camera Model

In [KHK10] there is an experimental comparison in simulated environment of the following camera models in motion relative motion estimation scheme very similar to ours:

1. Spherical approximation of an arrangement of 3 cameras similar to arrangement of cameras in Ladybug3 CM.
2. Single camera that can be modeled as pinhole CM.
3. An arrangement of 3 cameras as in 1., except that their optical centers coincide is a single point.

The error in imaging surface coordinates induced by approximation in item 1 was already discussed in section 1.2. Using terminology from that section, if $d_r > 7\,\mathrm{m}$, then the error introduced by the spherical approximation coincides with feature tracking error of 3 pixels. Inferring from experiments presented in [KHK10], it is reasonable to expect that spherical approximation will not behave much worse if d_r is as much as 10 times smaller.

This does not prohibit us to use spherical approximation, because of the fact that our design goals consider acceptable the possibility that the algorithm will not work in such small scene depths as it is expected that laser based odometry performs acceptably in such scenes.

There are other advantages to using spherical approximation. Namely that it is simpler then generalized CM and there are more methods and implementations for 5-point solvers. Additionally, as explained in the above in this section, using 5-point solver rather than 6-point solvers results in less iterations of RanSaC. Finally, given the fact that we use bundle adjustment (BA, Section 1.6), to refine our estimates, one could use motion estimation results obtained using spherical approximation to initialize BA which uses generalized CM. In such scheme distance of cameras from intersection of optical axes O_s would probably have to be considered as a global parameter, because the distance of the cameras from O_s is too small for reliable initial estimate of the scale.

Method Used for Model Estimation

Given the fact that there exist solvers usable in RanSaC scheme for both motion estimation and pose estimation problems with both generalized CM and central

CM case, we opted for RanSaC as a method for robust estimation as recommended in [FS12].

Regarding the choice of method for pose-graph edge relative transform problem. The implementation of motion estimation using 5-point algorithm is a must, because at various points in the VO process, 3D estimates may be unavailable or unreliable. If 3D landmark estimates are available, P3P would probably offer better initialization of scale estimate for BA. PnP is used as the main method of estimation in the VO method [NNB06]. Perhaps the most interesting approach to the estimation is described in [TPD08]. In this paper, it is suggested to use motion estimation method for estimation of rotation matrix and to use PnP method for estimation of translation vector. In the paper it is argued that motion estimation produces better estimates for rotation because firstly, it has many more and more evenly distributed datapoints and secondly, the feature-feature associations are direct observations whereas landmark 3D estimates are not. The case for estimating translation vector using PnP and not the method we use is then made by the better scale estimation capability of PnP.

It has already been established in the introduction of this thesis, that omnidirectional vision is far superior for motion estimation than single narrow field of view camera. The following is motivated by the fact that in our experiments and experiments of others, it was found that scale is very difficult to estimate as it is susceptible to drift [SMD10]. From the presentation of available solvers for central CM and generalized CM, it seem that there are algorithms of comparable quality available for severely generalized CM. Such algorithms have an advantage of being able to directly (i.e. from two cameras) estimate scale given that the camera optical centers are far enough apart (which ours are not).

1.4 Feature Detection and Matching

In this section, we discuss how we compute tentative feature-feature assciations based on appearance information. These are then used as input to pose-graph edge estimation (Section 1.3). We recall from Section 1.1 that feature-feature matches between poses A and B is determined by one-to-one map from features in image A to features in image B.

We decided to use common framework which provides unified interface for most methods used to solve our problem [Sze10]. In this interface is implemented by particular feature detector, feature description method and a similarity measure on the descriptors. The detector detects features in the image and some of their properties. One of the properties is feature response strength, which is a measure of feature quality. The descriptor describes useful information about feature appearence from the image. The similarity measure is a function defined on a pair of descriptors, its value is a metric that defines degree of similarity between two descriptors.

The best library for our purposes is OpenCV library (`http://opencv.willowgarage.com/wiki/`), which we use. This is because it provides the described interface and comes with many feature-matching methods implemented, but the only one that appeared to be suited for our purposes at the time was ORB [RRKB11]. This is mainly because it is fast enough and it is rotation-invariant (we will explain the invariance shortly). From our experiments confirmed that

it is satisfactory for our needs. Recently, many new feature detector-descriptor appeared and few of them are now directly in OpenCV and for most of them authors publish their implementations. For comparison and overview of these, we suggest reader consults [MM12].

The above statements indicate that choice of particular feature-matching method is not a design issue since they can be switched easily. There are two additional constraints, beyond taht of common interface, that needs to be satisfied. To explain them, we first introduce notion of *invariance of feature appearence to rigid body transform* (we loosely follow [Sze10]) of the landmark that corresponds to the feature. Common assumption used is that the landmark is plannar and thus the invariance to landmarks rigid body transform is modelled an affine transform in the image. Some feature-matching methods are invariant only to a subset of affine transforms (e.g. rotations, scaling, identity, or combinations of those). We now return to the two requirements of VO on the feature-matching method:

- One of them is **rotational invariance**. Because we are interesting only in features that are relatively far, the landmark poses change slowly except for rotations perpendicular to the ray connecting the landmark to the camera. This is manifested in the image by rotation and is caused by use of flippers on the robots when it overcomes high obstacles. Thus we have to require rotational invariance.

- The second requirement is **computational speed**, more precisely that of feature detection ad description. This disqalifies most of the methods.

It is common knowledge [FS12] that no particular method works well under all condition and thus it it desirable to combine multiple feature-matching methods, if resources permit it. Our design does not directly support it, but it can be fairly easily modified to do so.

1.4.1 Unguided Matching

In this section, we describe unguided matching component of Edge Builder. Unguided matching is preceded, for efficiency reasons, by what is commonly called non-maxima suppression. This is a process which reduces number of detected features and increases uniformity of distribution of features across the image. It is done by dividing image into local regions and selection for each region the best feature as measured by the feature response strength.

In this section we discuss how the above framework is used for matching. We used the approach suggested in this book [Sze10, Section 4.1.3]. It is based on *brute force matching*. Let A be set of features in one image and B set of features in the other. The method is as follows (repeated for each feature $a \in A$):

- For each pair (a, b), where and $b \in B$, a distance is computed using similarity measure. This is the most expensive step since the descriptors are quite large.

- For each feature in the source image, two closest features are identified.

- The ratio of the two distances is measured. If it is within tolerance (an ad-hoc parameter), the closest feature in the second image is accepted, otherwise it is rejected.

The reason for performing non-maxima suppression is to reduce number of computations of the similarity measure, which dominates the computational time consumed by the above method.

1.4.2 Guided Matching

The goal of guided matching is to match features that cannot be matched by the quided matching or to be used in situation where the RBT between the two poses involved in feature-feature matching is known. It is computationally much faster than guided matching.

The idea behind guided matching [DRMS07] is to use the pose estimate of the new pose and/or the landmark positions of features from the previous iterations. If both are available, the landmarks can be reprojected and the feature is searched for near the reprojection. If only the former is available, the missing depth is assumed to be infinite (this is equivalent to undoing only the rotational part of RBT).

The fact that search area for each feature is greatly reduced makes it efficient compared to unguided matching and compensates for the fact that many features cannot be matched by the unguided matching because there are many landmarks of identical appearence.

We compare the guided and unguided matching. Let w be image width in pixels, h be image height in pixels and $E(d)$ be feature density per pixel. Consider number of descriptor comparisons for brute force matching in unguided matching. This quantity amounts to $(E(d)wh)^2$. Now, consider a division of the image planes into regularly spaced grid, where each cell is of dimensions $\sqrt{w} \times \sqrt{h}$. Further, for each feature in the source image we know that it is located in one of the k of these cells in the target image. This means that under assumption that the image features are uniformly distributed across the destination image, the number of required comparisons amounts to $E(dwh(kd\sqrt{w}\sqrt{h})) = dwh(kE(d)\sqrt{w}\sqrt{h})) < (E(d)wh)^2$ for small k. For the image of size $1600x800$, the concrete values are $\sqrt{w} = 40$ and $\sqrt{h} = 28$. We use as large neighborhood as resources permit ($w = h = 60$ pixels, $k = 1$) — almost all possible feature-feature associations can found in this way.

Onw final interesting aspect of our guided matching is that we do not model landmark poses as RBTs, but as points in space. Thus feature matching based on knowledge of complete poses (as in [DRMS07]) cannot be done.

1.5 Feature-Landmark Association

The topic of this section is design of Landmark Manager module. Purpose of this module is twofold. First to maintain *feature-landmark association.* Second, computation of initial estimate of the landmark position for the bundle adjustment (Section 1.6). The first task consists determining which features are observations of which landmark, given feature-feature associations for edges in the pose-graph.

Landmark Manager is invoked by Edge Builder (Section 1.3) after feature-feature association is done for the edge that is being build.

For given feature (f, v), consider set of features $L_{(f,v)}$ defined as maximal set containing (f, v) such that $(f^{'}, v^{'}) \in L_{(f,v)}$ iff all the following is true:

- there exists $n > 0$
- there exists $((f_i, v_i) \in L_{(f,v)})$ for $i = 1 \cdots n$.
- $((f, v) = (f_1, v_1), (f_2, v_2) \cdots, (f_n, v_n) = (f^{'}, v^{'})$
- (f_i, v_i) is feature-feature associated with $(f_{i+1}, v_{i+1}))$ for $i = 1 \cdots n - 1$.

We call the set $L_{(f,v)}$ a *feature track* and in a naive implementation of Landmark Manager, the set $L_{(f,v)}$ could considered a set of feature observations of a landmark (i.e. be feature-landmark associated).

As discussed in the sections above, feature-feature matches are constrained by appearance similarity metric and geometrically, they only satisfy epipolar constraint given by the poses of nodes from which feature-feature constraint was constructed. This constraints the one of the features to to lie near the line given by the epipolar constraint and the other feature. Obviously, given the pose and a landmark position corresponding to the feature, any observation of that landmark can be constrained to lie around the landmark projection. In what follows, we detail how to identify such feature tracks and the corresponding landmark position that satisfy the mentioned geometric constraint. That is under the assumption of normally distributed landmark observation errors and (loosely speaking) accurately estimated camera poses for all nodes in the pose-graph. The motivation for verifying these additional constraint is the fact that landmark observations are required to be outlier-free by the bundle adjustment and the fact that the mentioned naive approach was found to unacceptable contain outliers. Our discussion of the topic proceeds firstly by discussing two major components of the design and then by presentation of the whole algorithm.

1.5.1 Preliminaries

Landmark-Feature Association Consistency under Available Geometric Constraints

We recall from Section 1.1 that our geometric model of observation of landmark l consists of landmark l modeled as a point in world CF. Observation of landmark l in camera c is then modeled as $\pi(T_{wc}(l))+\epsilon$, where π is a projection function given by the camera model (Section 1.2), T_{wc} a rigid body transform (RBT) from world CF to the camera CF and ϵ is measurement error, which is normally distributed with zero mean and covariance $\Sigma = \left(\begin{smallmatrix} a & 0 \\ 0 & a \end{smallmatrix}\right)$ for some $a \in \mathbb{R}$.

We now describe method that is referred to as *reprojection residual normality test*. Consider that we have set of features L, and we have precise estimates of $T_{wc(f)}$ for cameras $c(f)$ corresponding to the features $f \in L$, and we have a position estimate $p(l)$ of a landmark l. We wish to test the hypothesis that L are observations of landmark l under our model of observation. This consists of testing the statistical hypothesis that $S = \{(o(f) - \pi(T_{wc(f)}(p(l)))) \mid f \in L\}$ are samples drawn from distribution $N(0, \Sigma)$. A simple method for doing this, which

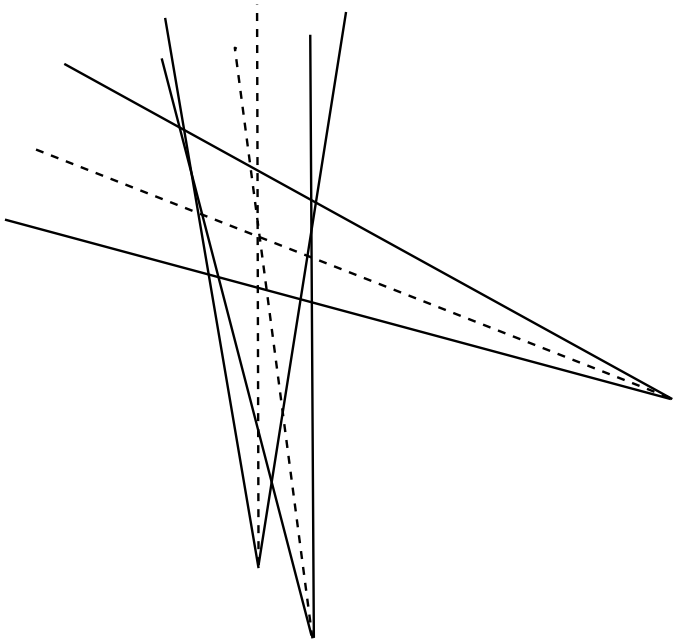


Figure 1.9: Mutual geometric consistency of landmark observation pairs (see text).

we use, is to do it by using reprojection error inlier function of Section 1.3.3 on each sample from S. It would be interesting to take advantage of the fact that cardinality of S makes it possible to use more advanced statistical methods for testing normality. It would be interesting to explore if these test would lead to some improvement.

Computing Landmark Position Estimate

Consider the following situation with the above model. We have set of features L, which are observation of landmark l. For each $f^i \in L$, we know the camera pose T^i_{wc} and we wish to compute an estimate of position $p(l)$ of landmark l.

The simplest and most efficient way we know of to compute estimate of position l is by selecting arbitrary pair of distinct features from L and using the two rays that correspond to the features and the poses of the nodes the features were observed from to triangulate the landmark position. Triangulation is done using Algorithm 3. This works to some extent, but it can be substantially improved upon. To show this, consider situation depicted in Figure 1.9. In this figure, we consider a 2D case of observing a landmark by three different rays. The true rays that observe the landmark, are drawn by dashed line. The area, where the measured observation can be with high probability (i.e., where it is considered an inlier by the method above), is bounded by the solid lines. If we consider arbitrary pair landmarks from the figure, the 3D estimate can be anywhere inside the intersection of the areas with highly probable observation rays. It can be seen that some pairs constrain the estimate of the landmark position higher than others and some pairs do produce estimates which would cause some inlier observations of L to be considered outliers by our method. We hypothesize that good pairs are those that have high disparity. *Disparity* of two rays (or the features corresponding to the rays) is defined as an angle between the two rays.

Motivated by this, we improve the naive estimation algorithm to compute the estimate by computing the estimate from all pairs of features and picking the first one that fits (decision made by the above method). If $|k|$ is high, we assume that the estimate $p(l)$ of the landmark l we already have from previous accepted set of observations for l is precise enough. This is motivated by the fact that robot cannot move arbitrarily fast which causes each next observation to be close. In more detail this procedure is presented as Algorithm 3.

Another, faster option, would be to pick the pair with highest disparity. This requires computing disparity for all pairs, but that should not be a problem since computing it is probably much more efficient than the triangulation.

Algorithm 4 Landmark Observation Set Consistency Test

Tests if landmark observation set L is consistent under our geometric landmark observation model and estimates landmark position. E_1 is a set of hints (known good landmark position estimates, if available) and k is a parameter of the method.

1. Let E be an empty ordered set.
2. Add E_1 into order set E with arbitrary order.
3. if $|L| > k$ continue with step 6.
4. Construct set $U \leftarrow \{(f_1, f_2) \mid f_1 \in L \wedge f_2 \in L \wedge f_1 \neq f_2\}$.
5. For each $p \in U$, construct triangulation using the feature pair p add the position estimate into E after all other elements already present in E.
6. For all e in ordered set E, do in ascending order:
   (a) For the set of observations L, test using the reprojection residual normality test described above, if L is consistent with e.
   (b) On test success, return $Yes(e)$
7. return No.

1.5.2 Algorithm For Maintaining Feature-Landmark Association

Let L be a set of features associated with landmark l. Such set uniquely represents landmark-feature association for l. Let $\mathcal{L}'$ be set of all L, the task of maintaining $\mathcal{L}'$ is then the task of feature-landmark association. One possibility to compute this set would be to check consistency of each feature track by the method described above and depending on the result, add such track into $\mathcal{L}'$ or discard it. We found experimentally that feature-feature associations contain too many wrongly associated features. Because of that, we would end up loosing a lot of potentionaly useful landmarks, which in turn decreases precision and increases risk of catastrophic failure in feature-deprived environments. Motivated by this, we decided attempt to to maintain $\mathcal{L}$ which is a minimal consistent subdivision

of $\mathcal{L}'$. More precisely, let F be the set of all features and let $L_(f, v)$ be defined for $(f, v) \in F$ as in the introduction. We want $\mathcal{L}$ to satisfy the following:

1. $(\forall(L_1 \in \mathcal{L}))(\forall(L_2 \in \mathcal{L}))(L_1 \neq L_2 \Rightarrow L_1 \cap L_2 = \emptyset)$
2. $(\forall((f, v) \in F))(\exists(\mathcal{L}_{(f,v)} \subseteq \mathcal{L}))((\bigcup_{K \in \mathcal{L}_{(f,v)}} K) = L_{(f,v)})$.
3. $\{L \mid L \in \mathcal{L} \wedge L \subseteq L_{(f,v)}\}$ is minimal in cardinality for all $(f, v) \in F$.
4. For all $L \in \mathcal{L}$, L is geometrically consistent in the sense of subsection 1.5.1.

In what follows the design of the algorithm for maintaining $\mathcal{L}$ and maintaining landmark position estimates for landmarks defined by elements of $\mathcal{L}$. We recall from previous text, that there are are two ways in which set of all feature-feature associations can change. One is by adding an edge into the pose-graph, the other is by removing a node from the pose graph. Note than in both cases each landmark can lose or gain at most one observation. We present a suboptimal algorithm (Algorithm 5) based on a greedy algorithm paradigm. It handles the case where a feature-feature association is added by determining if there exists a landmark with which the corresponding observation can be associated without compromising full landmark consistency. If this is the case than the observation is added to l, otherwise the uses o in an attempt to find new landmark.

Algorithm 5 Feature-Landmark Data Association Update

For each new feature-feature match $f = (o_1, o_2)$, update feature-landmark data associations as follows. k and l are parameters. There are three cases that are distinguished:

i **Both features in f are already associated with a landmark.** Let l_1 be landmark associated with observation o_1 and let l_2 be the landmark associated with observation o_2. If $l_1 \neq l_2$, then attempt to merge set of observations of landmarks l_1 and l_2 under new landmark with set of hints l_1, l_2 (Algorithm 4).

ii **One of the features is associated.** Without loss on generality, let o_1 be the associated as observation with landmark l_1. Attempt to merge set of observations $L_1 \cup \{o_2\}$ under new landmark with set of hints $\{l_1\}$ (Algorithm 4).

iii **None of the features are associated.**

(a) Let $L(n)$ be set of all feature tracks of n features that contain feature match f and do not contain any features associated with a landmark.

(b) For all feature tracks in $L(k)$, create a new landmark, if consistency test (Algorithm 4) succeeds and maximum disparity between pair of features in $L(k)$ is at least d.

(c) For all feature tracks in $L(l)$, create a new landmark if consistency test (Algorithm 4) succeeds.

We break our discussion of Algorithm 5 into the three cases. First, the case where an existing landmark is updated. Second, the case where a landmark is used for potential creation of new landmark. Third, the case of deletion of feature-feature association necessiated by pose-graph node removal.

Updating an Existing Landmark

When new pose-graph edge is added, each feature in the two nodes involved is contained in at most one new feature-feature association. We handle each feature-feature association as follows. There are two cases when the update of existing landmark is considered. First case is when both features involved in the feature-feature association are already associated with a landmark. The decision here is whether to merge the landmarks. Second case is when only one of the features is associated with a landmark. Both cases are handled in the fundamentally same way by attempting to find consistent estimate to the would-be new landmark by method of Algorithm 4 that was already discussed. This is the reason for the prerequisite that the rigid body transforms $T_{c_i w}$ for all poses in the graph are known. More precisely, the requirement is that only the relative transforms T_{ab} for all pairs of poses (a, b) involved in the invocation of Algorithm 4 are required to be well estimated. Given the fact that all node except possibly for the most recently added one were bundle adjusted (Section 1.6, this seems reasonable (we are not the only ones who think so [SMD10]).

There is one interesting detail about the case where both features are already associated with the same landmark. This means that there is a circle in the pose-graph and that each node in the pose graph that is part of the circle is also associated with the landmark and by the semantics of feature-feature and feature-landmark association, this means that for the set of features that correspond to the pose-graph circle each feature should be in exactly two feature-feature associations from the set of feature-feature associations involved in the pose-graph circle. This is not enforced by our algorithm and we do not consider it to be an issue, because the such inconsistency does not have direct effect on the bundle adjustment results.

New Landmark Initialization

At the beginning of this section, we claimed that $\mathcal{L}$ determines feature-landmark association. The actual set of landmarks is actually determined by subset of $\mathcal{L}$ defined as $\{L \mid L \in \mathcal{L} \wedge |L| \geq k(L)\}$, where $k(L)$ is minimum size of landmark observation set that is not same for all observation sets. Concretely, the values of $k(L)$ are set to $k(L) = 3$ for observations with sufficient disparity and to $k(L) = 4$ otherwise. These values were determined experimentally.

There are several reasons for choosing these values. One of the reasons we distinguish two classes according to disparity is that as discussed above, we hypothesize higher disparity estimates are more precise and thus more constrained. In our experiments some landmarks of only three features still contained outlier measurements which we believe is due to too permissive guided matching which produces lot of outliers that are highly probable to be consistent as determined by our test in Algorithm 5. Another reason is the fact that the Algorithm 3 is very likely to produce estimate with negative depths for landmarks that are far. Requiring more observations reduces probability that the landmark is incorrectly discarded. Yet another reason is the fact that lot of useful landmarks have only three observations, which is partially due to particular implementation of Keyframe Manager that we use.

Landmark Observation Deletion

When a node is deleted, the involved features are deleted from corresponding landmark observations. If this would cause a landmark l to have only one observation, the landmark l deleted. This implies that sets $L \in \mathcal{L}$ are no longer subsets of feature track sets $L_(f, v)$ in the new pose-graph where parts of the old pose-graph are forgotten.

1.6 Sliding Window Bundle Adjustment

We employ sliding window bundle adjustment to reduce drift and improve the pose estimates suggested in [FS12]. Bundle adjustment can be formulated as a non-linear lest squares problem on manifolds as is done in g2o framework [KGS+11] that we use. The term *sliding window bundle adjustment* refers to the fact that bundle adjustment is performed on k closest poses to the last one in the pose-graph. In this section we define objectives we want to achieve and describe our formulation of the problems in term of the requirements described in the manual [GKSK12] to g2o framework. For introduction on how algorithms for NLSOM operate, we refer the reader to [GKSB10] and [Her08].

We have chosen *g2o* [KGS+11], because it is well documented [GKSK12], easy to use, well coded, and used by others (e.g. [SDMK11]). It is general in terms of its interface and subclasses of nonlinear least squares optimalization problems that it is able to solve efficiently.

1.6.1 Non-Linear Least Square Optimalization on Manifolds

In the following, we define non-linear least square optimalization on manifolds [Her08]. We have parameters $X = (x_1, x_2, ..., x_n) \in M$ that we wish to optimize. We are given set observations $Z = z_1, z_2, ..., z_m, z_i \in \mathcal{R}^{k_i}$, where the i^{th} observation is assumed to be determined by function $f_i \colon X \to \mathcal{R}^{k_i}$ up to a Gaussian noise. That is the measurement i is drawn from normal distribution with mean $f_i(X)$ and covariance matrix Σ (which is given) and the measurements are independent given $f_i(X)$. This can be expressed by probability density $p_i(Z_i = z_i|X = x)$ which is called observation model in context of SLAM.

The goal is to determine local minimum of maximum likelihood function

$$x^* = \text{argmax}_x \prod_i p_i(Z_i \mid X = x) = \text{argmin}_x \sum_i -log(p_i(Z_i \mid X = x)$$

Common algorithms that find solutions for the problem are Gauss-Newton, Levenberg-Marquardt and Conjugate Gradient methods. They proceed by starting with given initial estimate x_0 and in each iteration k deriving estimate x_{k+1} from estimate x_k of the previous iteration and the gradient of the function at x_k. This means that on convergence x_k is only local optimum. Thus in order to find global optimum, the initial guess x_0, which is provided as an input should be close enough to the optimum.

1.6.2 Optimalization Criteria for Our Problem

In this section we describe optimalization criteria used in context of $g2o$. In $g2o$ NLSOM is represented as a hyper-graph, where nodes correspond to parameters x_i and hyper-edges represent observation models in form of probability density $p_i(Z_i = z_i|X = x)$, where the hyper-edge is connected to the nodes that p_i is dependent on. The probability density is specified in terms of function f_i and an information matrix $\Omega_i = \Sigma_i^{-1}$.

Node Parametrization We have two kinds of nodes – landmark nodes and camera pose nodes. Landmark nodes are represented as points in 3D space and parametrized as euclidian coordinates in $\mathcal{R}^n$ relative to world coordinate frame. Landmark increments have the same parametrization.

Camera poses are represented as rigid body transforms from world frame to coordinate frame of the camera. They are parametrized in the domain of $SE(3)$ group. The increments are parametrized in domain of $se(3)$ group and the increment operator is realized by mapping increment around identity to its corresponding $SE(3)$ element using exponential map [MSKS10]. The resulting $SE3$ element is then multiplied by the computed $SE(3)$ element (rigid body transform concatenation). This is minimal parametrization (with respect to degrees of freedom) in eucleidian space. There are two common minimal parametrizations in euclidian spaces for increments – the $se3$ group and (tq) where t is translation vector and q is unit quaternion. We opted for the former because the two were experimentally evaluated in context of bundle adjustment in [KGS+11] and the $se3$ came out slightly better in terms of speed of convergence.

Observations Only pure observation in our task are observations of landmarks that are manifested by detected features in each camera pose (additionally estimated pose-to-pose transforms could be considered). Feature measurements are represented in image coordinates. Problem of data association was already treated in Section 1.5. At this point in computation, the data associations are available. The covariance of information matrix is set to identity matrices. The Gaussian distribution of measurement is common assumption in the literature in our setting as is independence of observation given model parameters.

Observation Model and Error Function As an observation model is obviously given by the projection of landmark corresponding to given observation (see section refsec:camera-models for discussion of camera models). The error function used is the reprojection error function from Section 1.3. The error function adjusts for singularities at image borders.

Initialization of Parameters This is the topic of Section 1.3 and Section 1.5.

2. Experimental Results

In this section, we compare our VO system with other means of odometry estimation. It will be shown that rotational part of the estimated RBTs by our system is superior to results obtained from INSO. Even though translational part of the RBTs is not as well estimated and contains gross errors, the result can be actually useful, if results from INSO and VO are merged as was initial planed. This mege would be necessary anyway, because the sampeling rate of VO is too slow for some applications.

We tested our system on four datasets. All of the datasets use panoramic images of resolution 1600×800, which are processed at a rate of about 2-3 frames per second. We compare the results obtained using VO on each dataset with one of the following methods.

Reference Tracking System (RTS) In this method, robot trajectory is measured by using an external camera to track a pair of markers attached to the robot. This method is reported to have localization accuracy of 15 ± 13cm and heading accuracy of 3.8 ± 2.7deg. The sampeling frequency of this method greather than 20Hz. For further information about the method, consult information presented in [KBK12].

Inertial Navigation System / Odometry (INSO) The method computes odometry by fusing outputs from odometries obbtained from inertial navigation system and wheel odometry. The sampeling frequency for this method is grather than 100Hz. For more information about INSO, see `https://cw.felk.cvut.cz/doku.php/misc/projects/nifti/sw/ins` and `https://cw.felk.cvut.cz/doku.php/misc/projects/nifti/sw/inso`.

This method is locally more precise than our VO system and thus is still useful for evaluating datasets where RTS is not available.

The comparison is done by aligning the trajectories from VO and one of the reference methods decribed above. The aligned trajectories are then visually compared, the poses where the trajectories differ are identified and VO log is then conulted to arrive at an explanation of what is the cause. The trajectory alignment method is as follows:

1. for each pose computed by VO, a corresponding pose from the reference method is selceted as the one that has the closest time-stamp.

2. For each VO pose v, a transform T^A that aligns VO track by use of similarity transform to the corresponding pose of the reference method m is computed as $T^A = T_{vw} S T_{wm}$, where S is appropriate scale transform.

3. From all T^A, the transform that minimalizes maximum distance between corresponding poses of VO and the reference method is selected.

Usually, no single good alignment exists, and so we divide the dataset to few sets of consecutive poses. For each such set an alignment is produced in such a way that the above measure is computed only on this set. An example of this is

figures 2.1 and 2.2. The reason for the fact that we need multiple alignments is that our VO system poorly estimates scale. To illustrate this for some datasets, we plot all poses colored by the scale factor that would have been used, if a given pose was used for the alignment. An example of this is Figure 2.3.

In the following, we describe the four datasets that we evaluated and present the results using the above method. Detailed analysis of the local failures is then presented in the following section:

Yard Dataset Longer dataset of length 46.6m in a yard placed inside a city block. Both INSO and RTS data are available for this dataset. The results are of the similar quality as in Street Loop Dataset, we present them in figures 2.1 and 2.2.

Rail Car Dataset Long dataset recorded in a railyard. Robot is driven around a large rail car and almost closes a loop. Only INSO data are available and length is of this track is approx. 100m long according to the INSO.

Results are presented in figures 2.4 and 2.6. From Figure 2.3, it is obvious that our VO implementation drifts in scale.

Rail Following Dataset The longest dataset, which is recorded in a railyard. Robot is driven alog between straight rails and back, then it is driven out of the rails, approximately at the place it started and finaly along the side of the same pair of rails. Again, only INSO data are available and the length calculated from INSO data is approx. 370m.

Results in Figure 2.7 definitively demonstrate that VO, even in its present form can be sucessfuly used to correct rotational drift produced by INSO. Figures 2.8 and 2.10 visualize scale issues.

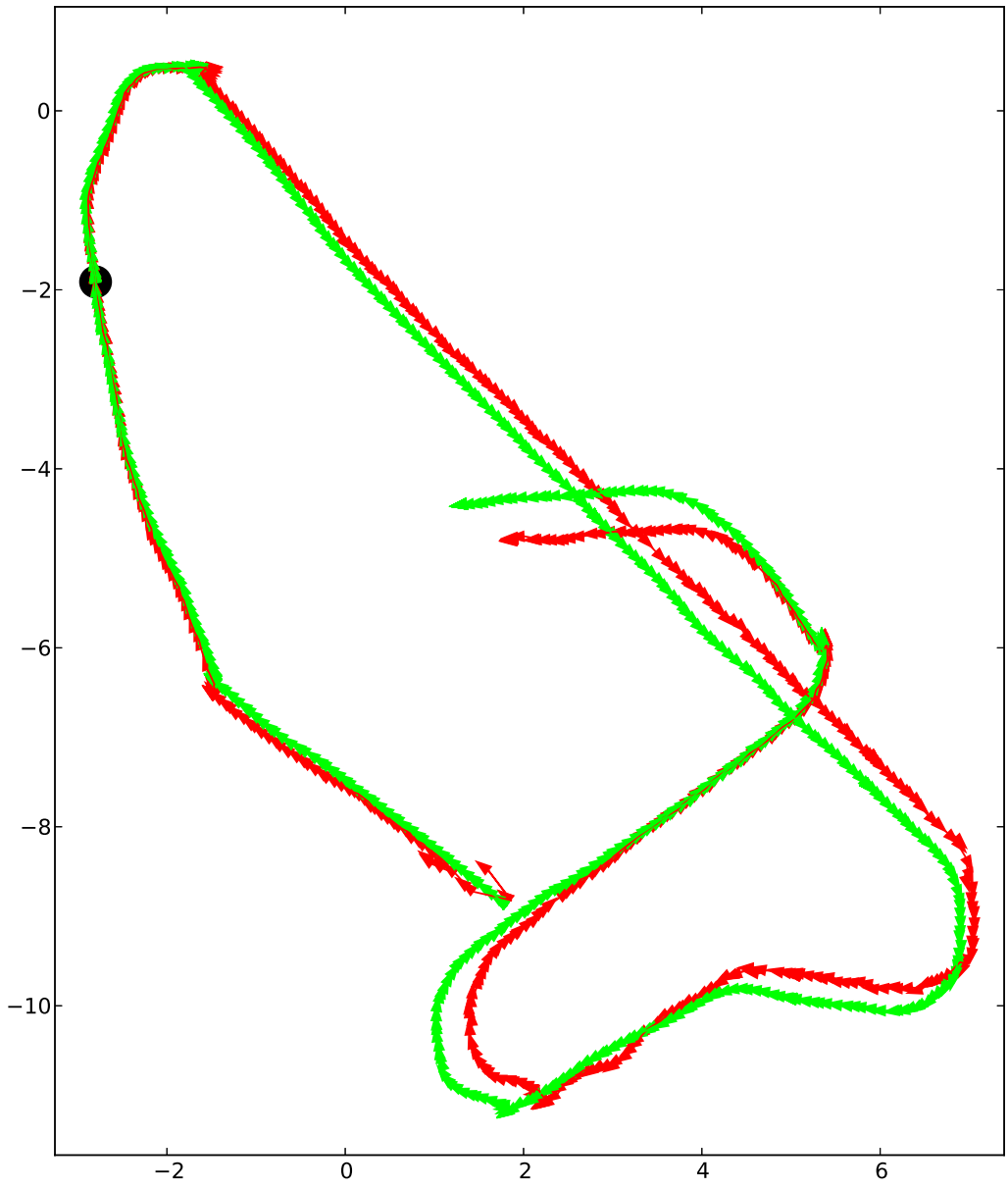
Indoor Dataset This is small dataset where robot drives in a corridor and around stairwell of an office building. In Figure 2.11, there are two poses displayed, were VO severly misestimates pose. This is caused by the fact that our camera model was not designed to operate indoors (parts of image with high angle to closest optical axis are unusable). Additionally, there is some glass which reflects surrounding scene.

2.1 Discussion of the Scale Drift of our System

It is well known that visual odometry drifts in scale, however we believe that the drift seen e.g. in Figure 2.3 can be improved uppon. There are two improvements that we believe would help to reduce the drift. They should be tried in the order they are presented in.

In our testing we have learned that with a few exception all landmarks are lost after two meters of trajectory length, and that increasing this length improoves scale estimates. We belive that one of the causes of this is that apearance of landmarks changes so much that they cannot be reassociated anymore and then they are started as new landmarks. We came-up with these two possible causes:

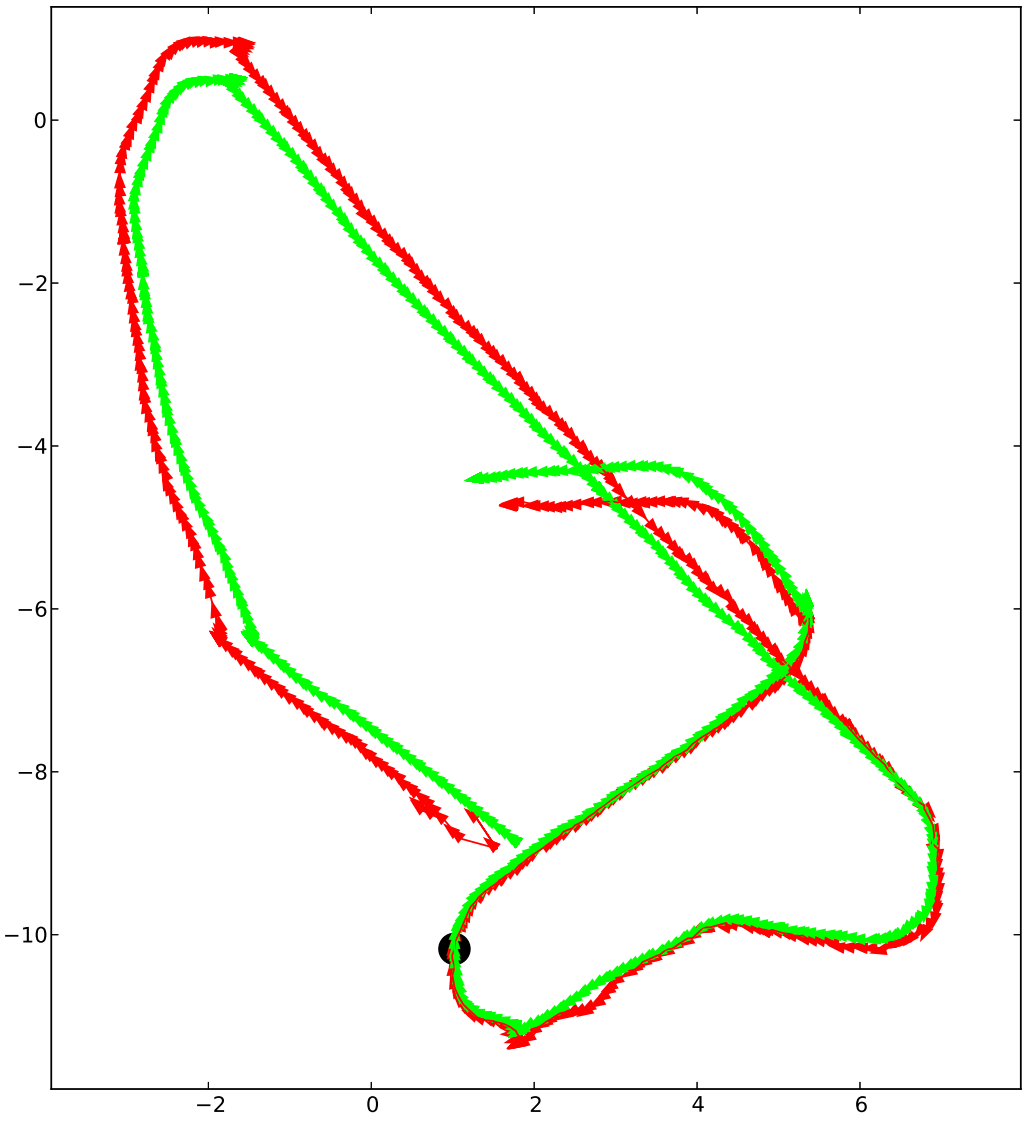
- We use feature detector/descriptor that is not scale invariant and thus landmarks are lost if scale changes too much between keyframes.



(a) Trajectory Alignment.

(b) Picture for the pose marked by the black dot.

Figure 2.1: Yard Dataset. RTS trajectory (green) aligned with VO trajectory (red) ad the pose marked by the black dot.



(a) Trajectory Alignment.

(b) Picture for the pose marked by the black dot.

Figure 2.2: Yard Dataset. RTS trajectory (green) aligned with VO trajectory (red) ad the pose marked by the black dot.

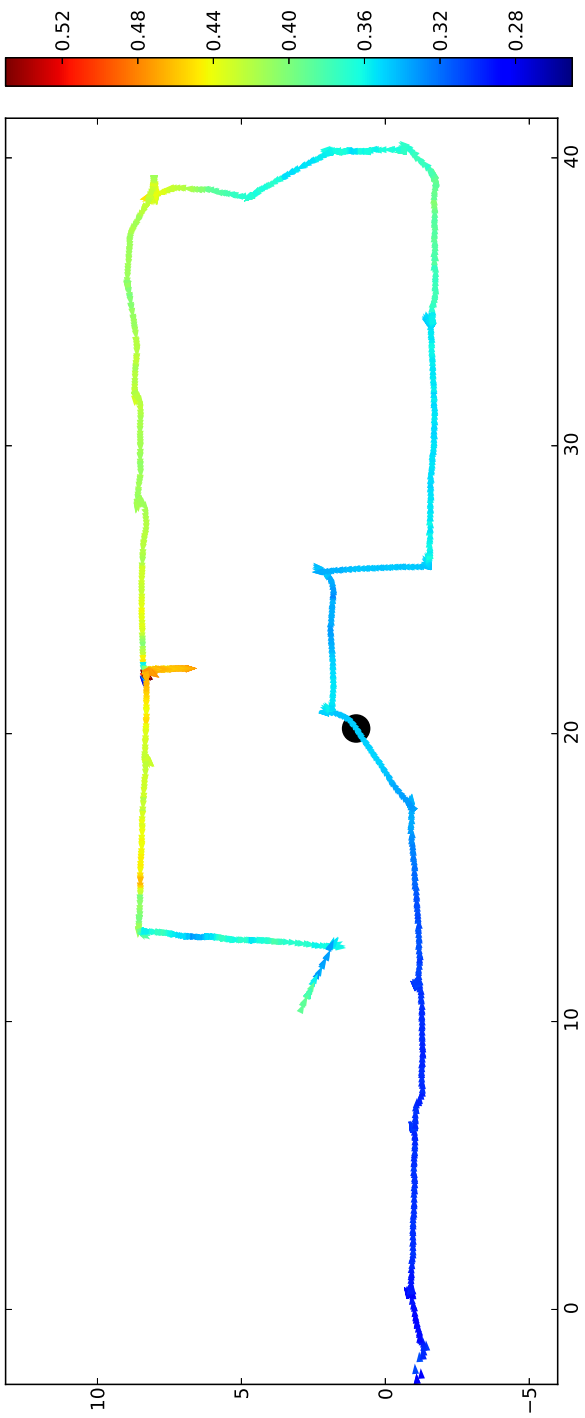


Figure 2.3: VO trajectory aligned to INSO trajectory at the pose marked by the black dot. INSO trajectory is not plotted. Poses are colored by the scale factor that would have been used, if a given pose was used for alignment. Units are in meters and axes do have same scale.

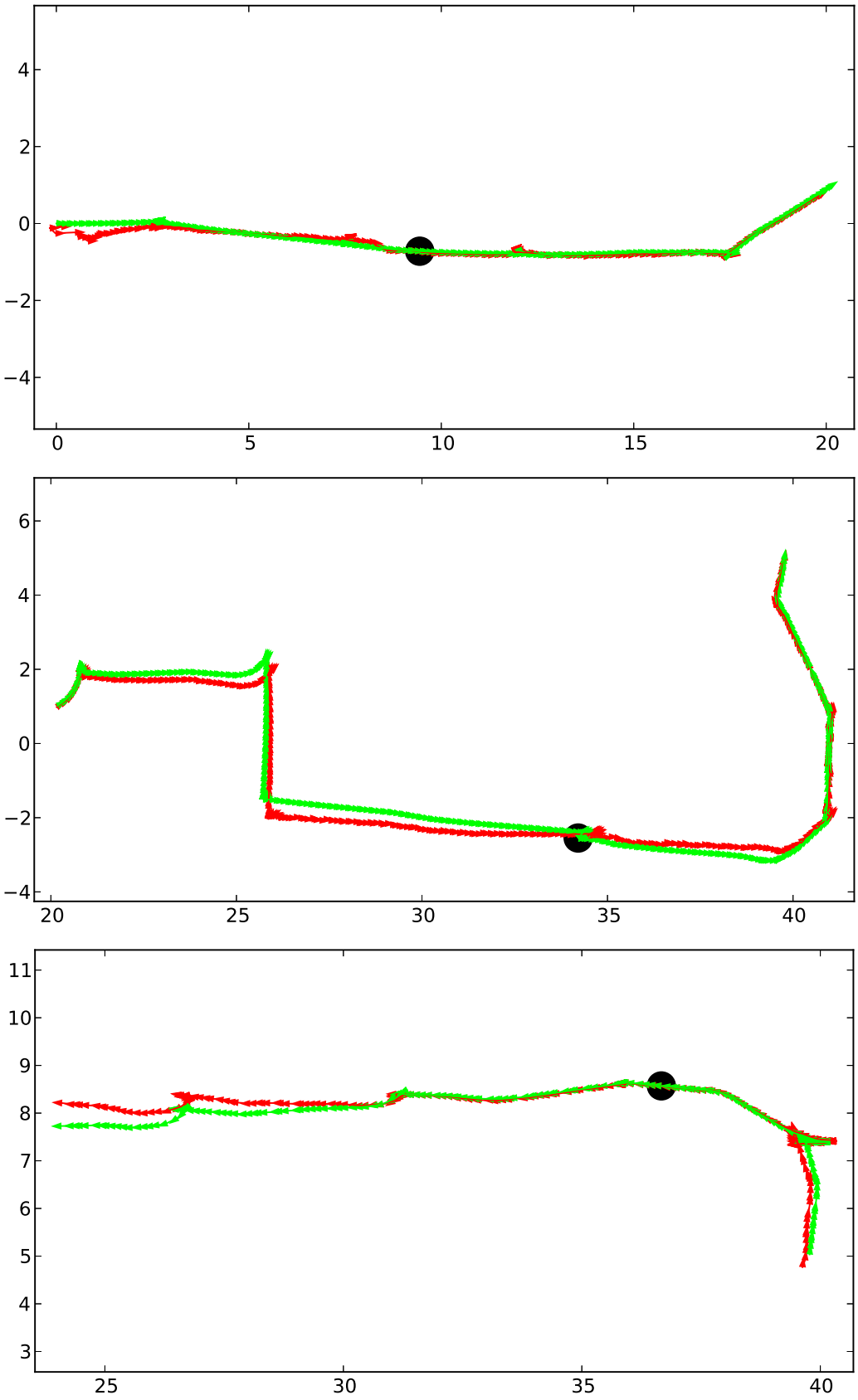
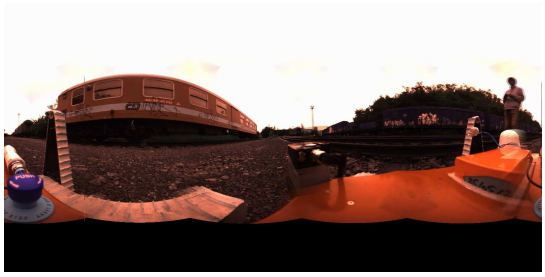


Figure 2.4: Alignments of VO trajectory to INSO trajectory presented in order of increasing timestamp of the alignment pose. Alignments, first to third.



(a) first alignment picture

(b) second alignment picture

(c) third alignment picture

Figure 2.5: Vagon Dataset. Robot camera pictures taken at the alignments of Figure 2.4.

- Panoramic image distorts the image patches used for computation of the descriptor and thus feature appearence changes as it moves around the image plane.

Another cause of this can be feature occlusions. Thus the second improvement would be to improve Keyframe Manager component of the system as follows. In the current implementation, the pose-graph is a tree. This means that once a feature is lost in a single key-frame, it is never reaquired as an observation of the original landmark. This occurs quite often and could be improved by implementing what was introduced in Section 1.1.4 as edge construction with known RBT and using it to make keyframe manager to do mini loop-closures.

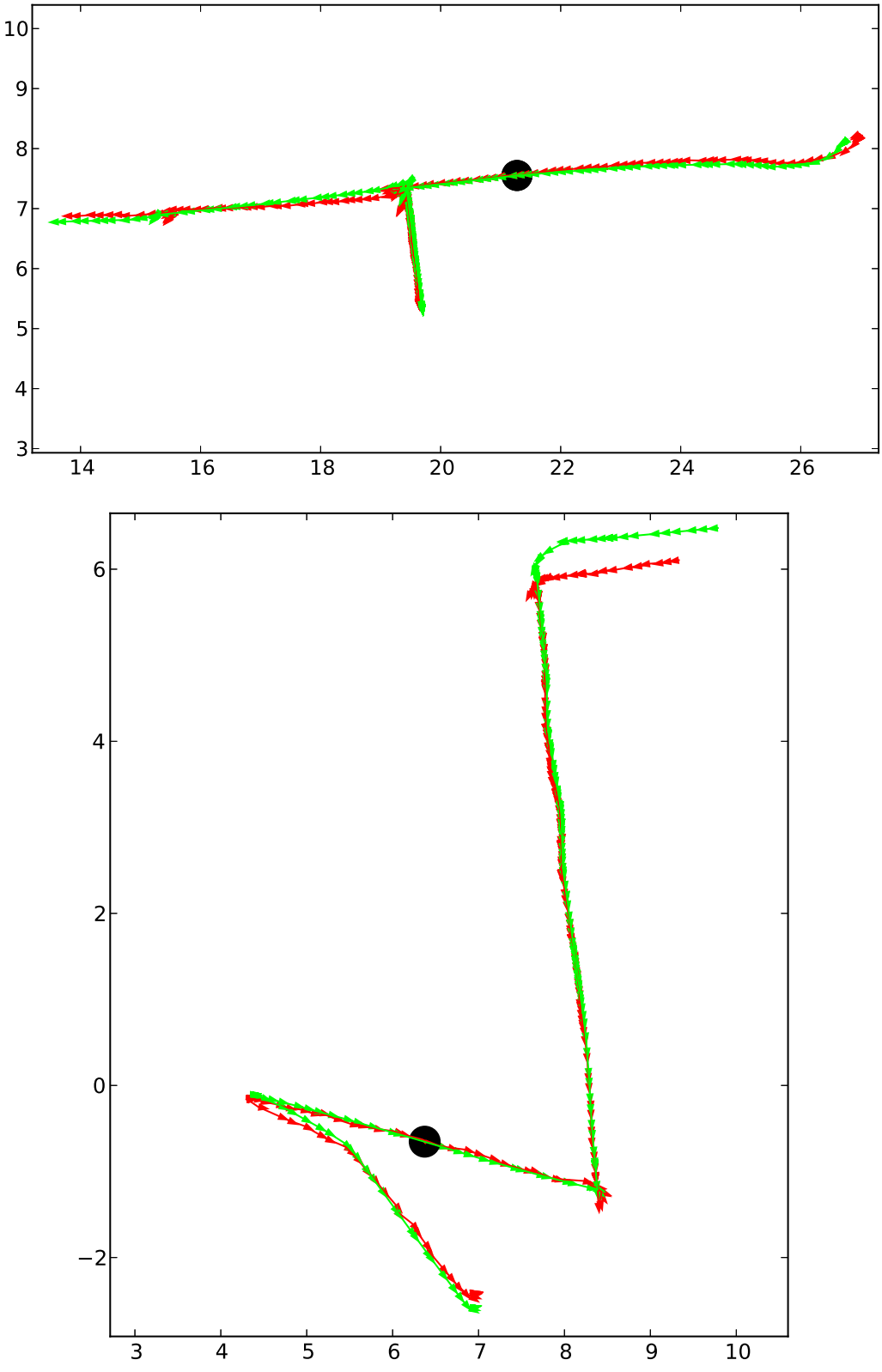
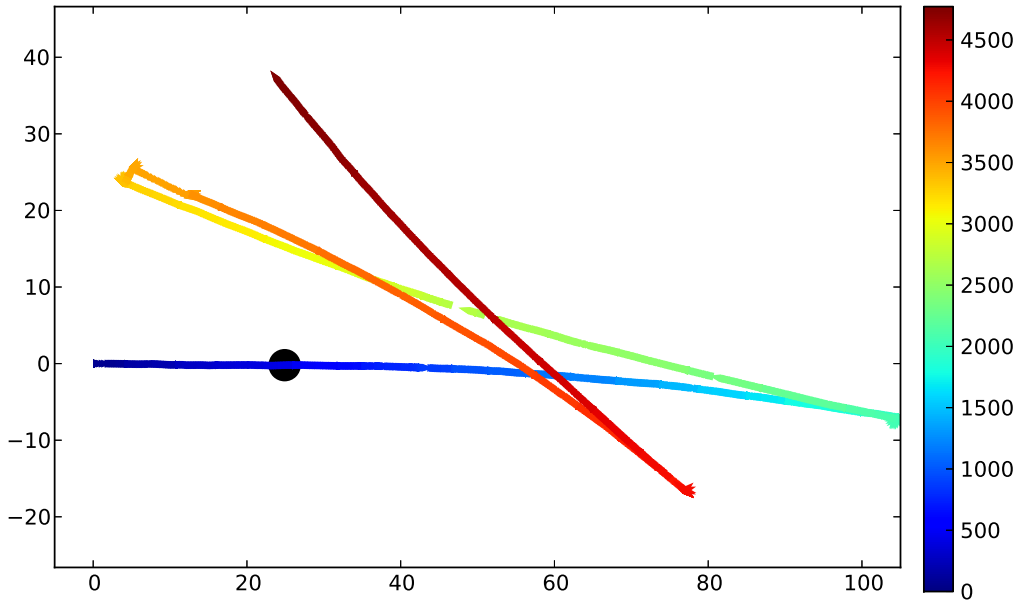
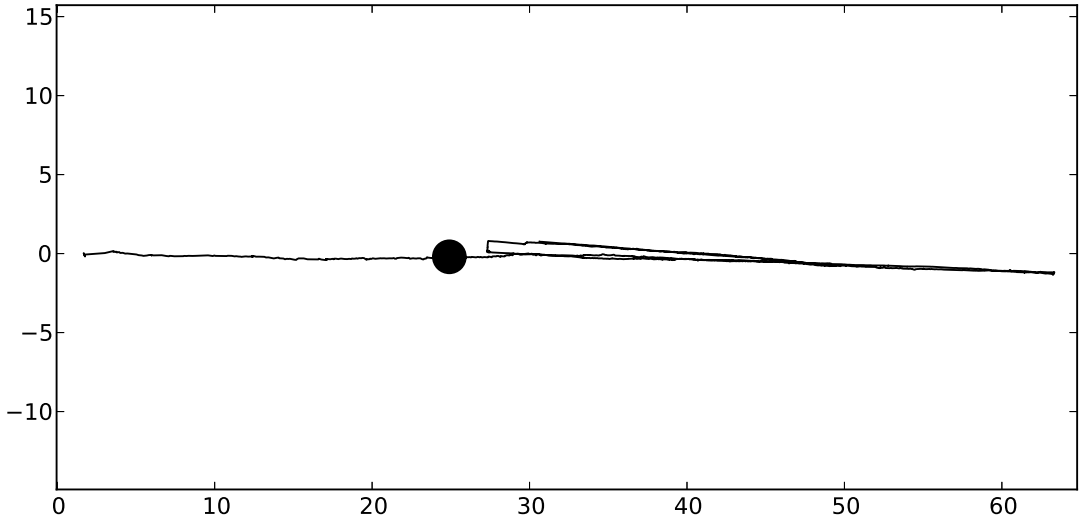


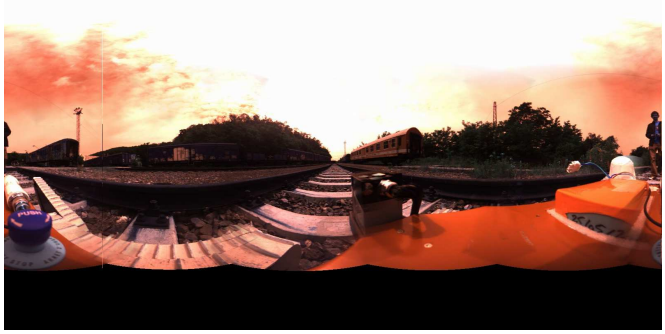
Figure 2.6: Alignments of VO trajectory to INSO trajectory presented in order of increasing timestamp of the alignment pose. Alignments, fourth and fifth.



(a) INSO trajectory. Color indicates time ordering of the poses. 0 corresponds to the pose with smallest timestamp.



(b) VO trajectory aligned with INSO trajectory. INSO trajectory is displayed in (a).



(c) Image from the dataset.

Figure 2.7: Rail Following dataset. Units are in meters and axes in each graph have the same scale. The black dot marks pose which was used to align VO trajectory to INSO trajectory.

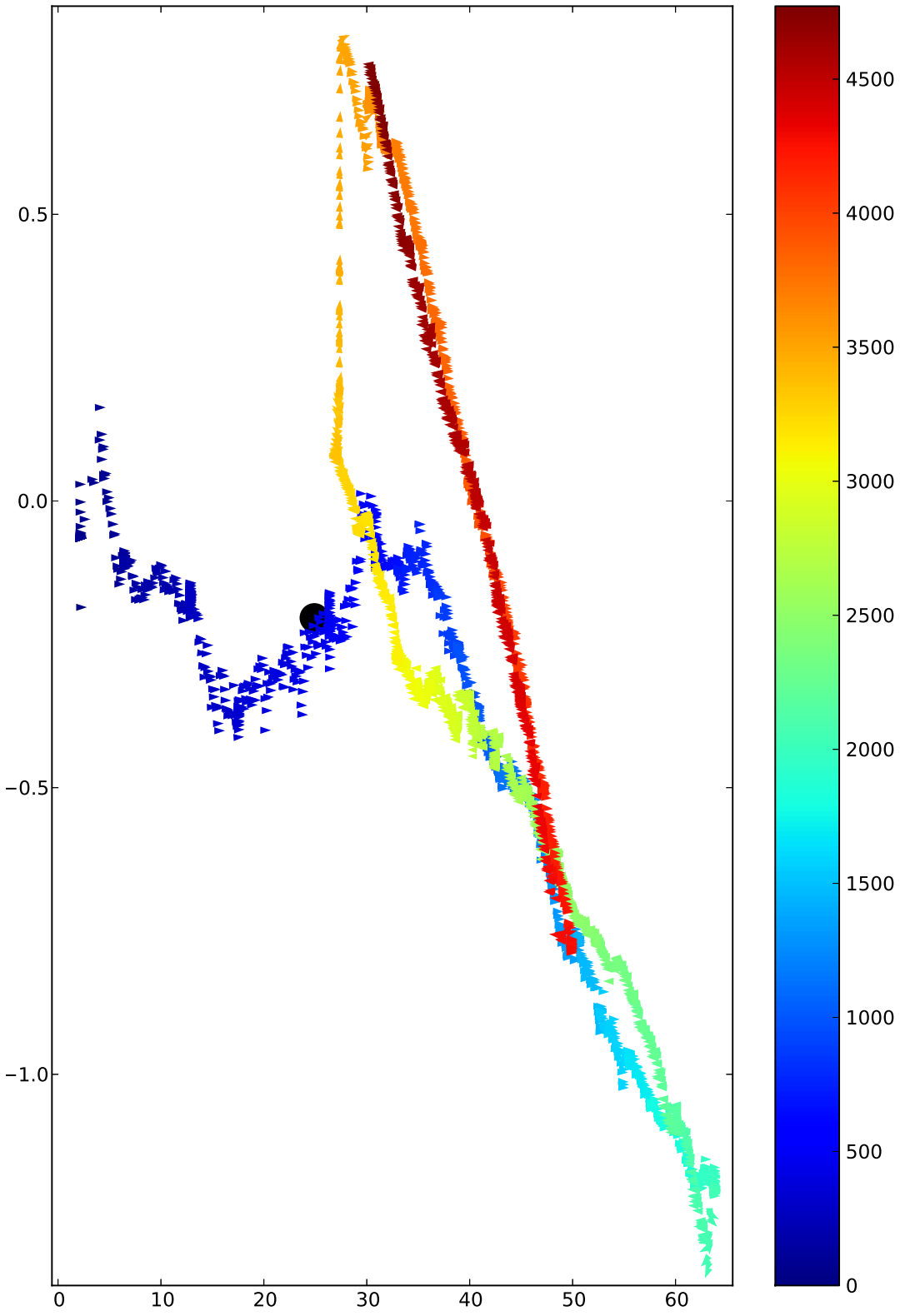


Figure 2.8: Rail Following dataset. Units are in meters and axes in each graph do not have the same scale. The black dot marks pose which was used to align VO trajectory to INSO trajectory. Color indicates time ordering of the poses. Value 0 corresponds to the pose with smallest timestamp.

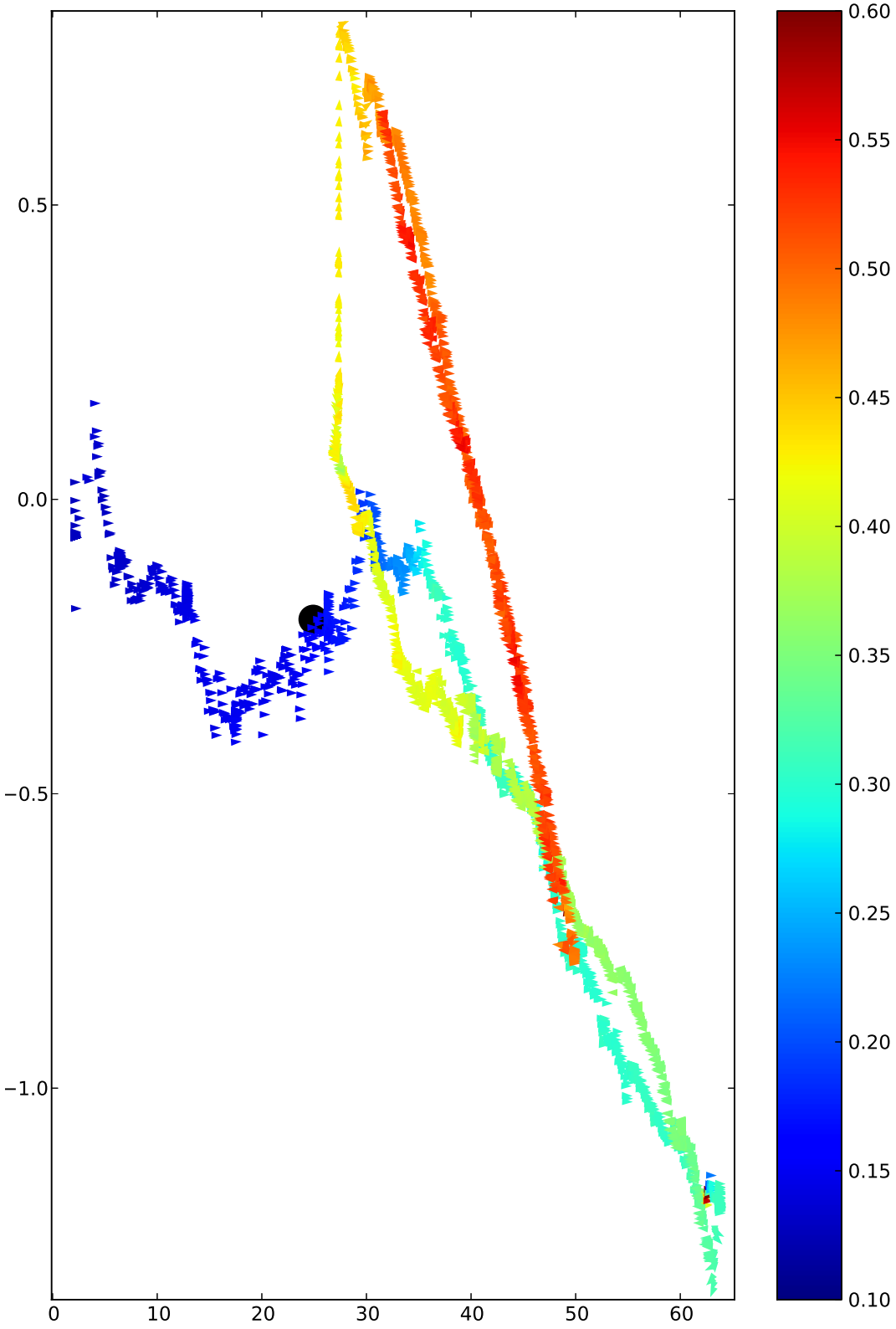
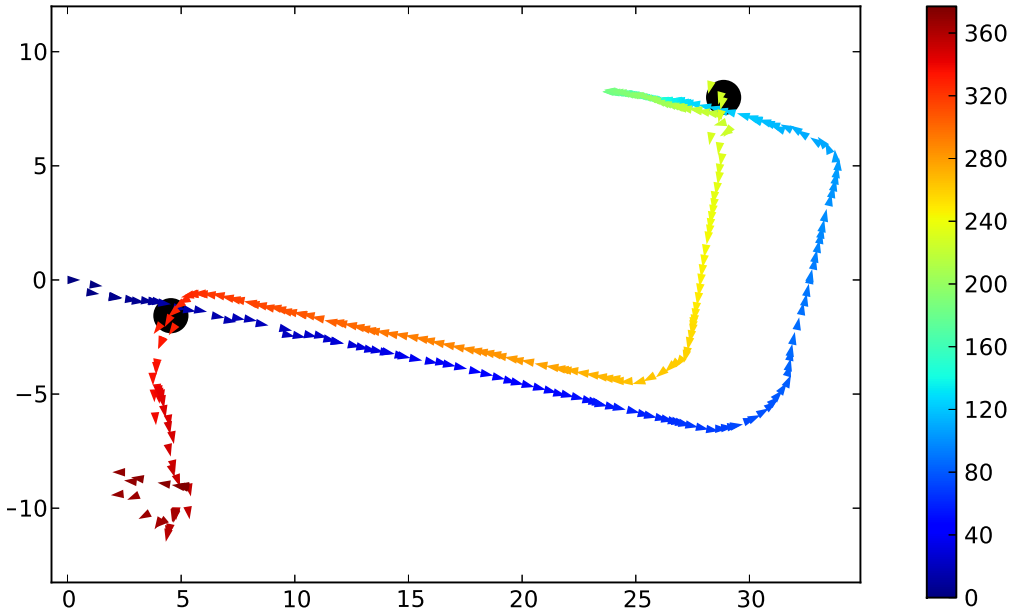


Figure 2.9: INSO trajectory.

Figure 2.10: Rail Following dataset. VO trajectory aligned with INSO trajectory at the pose marked by the black dot. Poses are colored by the scale factor that would have been used, if a given pose was used for alignment. Units are in meters and axes do not have same scale.



(a) Scale of this trajectory is undetermined. The pose colors correspond to timestamps.

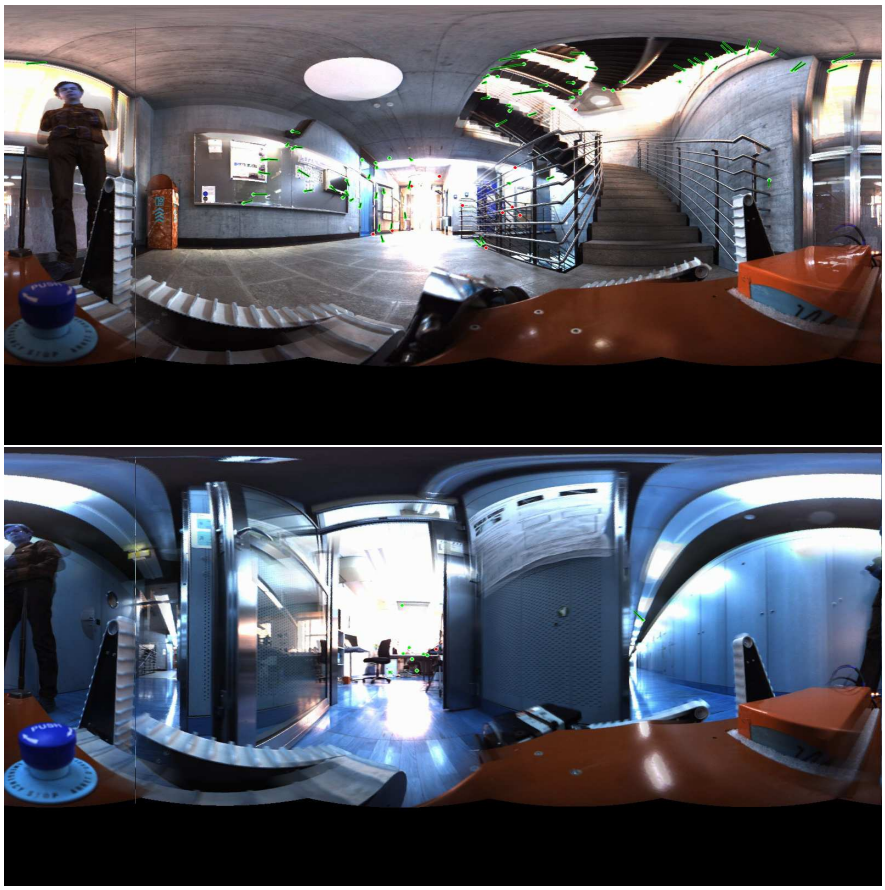


Figure 2.11: Indoor Dataset. The two images correspond to the poses marked by the black dot. The upper image corresponds to the upper dot.

Conclusion

The motivation behind developing visual odometry for the robot was to use specific strengths of omnidirectional image sensing to complement odometries that are already implemented for the robot. The main strengths of omnidirectional sensing in the context of visual odometry are mainly the superiority of rotation estimates and the fact that it is suited for outdoor environments, where scene object are typically far.

As it was already demonstrated in Chapter 2, the estimates in rotation are superior to that of INSO. The main problem with our implementation is the poor estimation of scale, which limits VO's usefulness a a standalone solution. The ways that could correct the scale drift were already described in Section 2.

Currently, a system is being built to integrate the odometries from all three sources. Under such system the rotation estimates produced by the VO could be used to improve estimates from other sources. The scale drift of the VO system poses a problem, if the translation parts of the estimates are to be used. In closing, we would like to note that there will always situations where VO inherently fails (e.g. dark environments). Thus, it inevitable to rely on multiple sources for odometry.

Bibliography

[DRMS07] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, 2007.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[FS12] F. Fraundorfer and D. Scaramuzza. Visual odometry : Part ii: Matching, robustness, optimization, and applications. *Robotics Automation Magazine, IEEE*, 19(2):78–90, june 2012.

[GAPP07] R. Goecke, A. Asthana, N. Pettersson, and L. Petersson. Visual Vehicle Egomotion Estimation using the Fourier-Mellin Transform. In *Proceedings of the 2007 IEEE Intelligent Vehicles Symposium IV'07*, pages 450–455, Istanbul, Turkey, June 2007. IEEE.

[GKSB10] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based slam. *Intelligent Transportation Systems Magazine, IEEE*, 2(4):31–43, 2010.

[GKSK12] G. Grisetti, R. Kummerle, H. Strasdat, and K. Konolige. g2o: A general framework for (hyper) graph optimization. jan. 2012. Last checked 2012.07.14.

[Her08] Christoph Hertzberg. A framework for sparse, non-linear least squares problems on manifolds, 2008. Supervisor: Udo Frese.

[HWB+11] C. Hertzberg, R. Wagner, O. Birbach, T. Hammer, and U. Frese. Experiences in building a visual slam system from open source components. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2644–2651. IEEE, 2011.

[HZ04] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, New York, second edition, 2004.

[KBK12] V. Kubelka, V. Burian, and P. Kafka. Reference tracking system for a mobile skid steer robot. Technical report, Center for Machine Perception, CTU, May 2012. Available at https://cw.felk.cvut.cz/lib/exe/fetch.php/misc/projects/nifti/sw/-reference_tech_report.pdf.

[KGS+11] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613. IEEE, 2011.

[KHK10] Jun-Sik Kim, Myung Hwangbo, and Takeo Kanade. Spherical approximation for multiple cameras in motion estimation: Its applicability and advantages. *Computer Vision and Image Understanding*, 114(10):1068 – 1083, 2010.

[KSS11] L. Kneip, D. Scaramuzza, and R. Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2969–2976, june 2011.

[LH06] Hongdong Li and R. Hartley. Five-point motion estimation made easy. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 630–633, 2006.

[MM12] O. Miksik and K. Mikolajczyk. Evaluation of local detectors and descriptors for fast feature matching. In *Pattern Recognition (ICPR 2012), 21st International Conference on*, pages 2681–2684. IEEE, 2012.

[MSKS10] Y. Ma, S. Soatto, J. Kosecká, and S.S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*, volume 26 of *Interdisciplinary Applied Mathematics Series*. Springer, New York, 2010.

[MW08] M.J. Milford and G.F. Wyeth. Single camera vision-only slam on a suburban road network. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3684–3689, may 2008.

[NNB06] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, 2006.

[NS07] D. Nistér and H. Stewenius. A minimal solution to the generalised 3-point pose problem. *Journal of Mathematical Imaging and Vision*, 27(1):67–79, 2007.

[Ple03] R. Pless. Using many cameras as one. In *Computer Vision and Pattern Recognition (CVPR), 2003 IEEE Computer Society Conference on*, volume 2, pages 587–593. IEEE, 2003.

[RRKB11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564 –2571, nov. 2011.

[SDMK11] Hauke Strasdat, Andrew J. Davison, J.M.M. Montiel, and Kurt Konolige. Double window optimisation for constant time visual slam. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2352–2359, nov. 2011.

[SF11] D. Scaramuzza and F. Fraundorfer. Visual odometry [tutorial]. *Robotics Automation Magazine, IEEE*, 18(4):80–92, dec. 2011.

[SMD10] H. Strasdat, JMM Montiel, and A.J. Davison. Scale drift-aware large scale monocular slam. In *Robotics: Science and Sytems (RSS), In Proceedings of*, Zaragoza, Spain, June 2010.

[SNOÅ05] H. Stewénius, D. Nistér, M. Oskarsson, and K. Åström. Solutions to minimal generalized relative pose problems. In *Workshop on Omnidirectional Vision*, Beijing China, October 2005.

[SRT+11] P. Sturm, S. Ramalingam, J.P. Tardif, S. Gasparini, and J. Barreto. *Camera Models and Fundamental Concepts Used in Geometric Computer Vision*, volume 6 of *Foundations and trends in computer graphics and vision*. Now Publishers, 2011.

[Sze10] R. Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. Springer, 2010.

[TPD08] J.P. Tardif, Y. Pavlidis, and K. Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2531–2538. IEEE, 2008.

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

VO	visual odometry
BA	bundle adjustment
NLSM	non-linear least squares on manifolds
RanSaC	random saple consensus
CM	camera model
CF	coordinate frame
RBT	rigid body transform
PnP	perspective from n points
RTS	reference tracking system
INSO	inertial navigation system / odometry